A Prototype Optical Tracking System: Investigation and Development

Crispin D. Lovell-Smith

Department of Electrical and Computer Engineering

A thesis presented for the degree of
Master of Electrical and Computer Engineering

University of Canterbury
Christchurch, New Zealand
June 2009

Abstract

Tracking of an object in six degrees-of-freedom (DOF) produces a position and orientation estimate of the object as it moves in 3D space. This thesis investigates the design and implementation of a prototype optical 6 DOF tracking system. Although optical scanners potentially have issues regarding occlusion they have advantages over electromagnetic scanners in that they can be used without distortion near ferromagnetic materials and can have large working volumes.

This thesis focuses on the design of a small camera module named the 'Black Spot' that forms part of the overall tracking system. This module is capable of tracking the locations of up to 27 LED markers at 60 frames/s as the module moves in space. These markers provide fixed reference points that are utilised by the tracking system. A number of these modules will, in future revisions of the system, be clustered closely together forming a tracking hub. In this research this hub has been partially implemented in software on a PC. This software implements a 'pose estimation' algorithm that iteratively refines the location and position of the camera modules.

Results from testing three Black Spot modules indicates that the locations of the LED markers can be determined very precisely using a centroid calculation. Standard deviations of better than 0.01 pixels have been recorded using these modules. The pose estimation algorithm has been tested revealing the need for a better minimisation algorithm. It is recommended that a bundle adjustment algorithm is used in the future to refine the world model used by the hub. The calibration of the system is a task for future research.

Acknowledgements

Many people have helped me throughout this project and I would like to acknowledge them. Firstly, I would like to thank my supervisors: Professor Phil Bones and Dr Michael Hayes at the University of Canterbury, and Dr Rachel Johnston at ARANZ Scanning Ltd. Throughout the two years that I have spent on this project they have had significant input. Despite their busy schedules they have found time to proof read many drafts of my thesis in recent months.

In particular, I would like to thank Phil for his regular meetings in which we spent time with Michael pondering the best ways to design the tracking system. Thank you for organising the two trips to the IVCNZ conferences in 2007 and 2008, and for writing the Black Spot paper for the most recent IVCNZ conference.

I would like to thank Michael for his help in the hardware and firmware development of the Black Spot module. Michael is enthusiastic towards embedded systems and was always willing to help me tackle 'gnarly' bugs (as he calls them) in the early stages of the Black Spot development. Michael suggested that I switch to Open Source tools early in the project and provided libraries of his own code for low level Blackfin routines.

Rachel has proof read many chapters of my thesis at a rate at which I have struggled to keep up with. Despite officially working part time she is extremely busy and I suspect she spends much of her 'free time' staying up to date with her work. Rachel's suggestions have been invaluable. I would like to thank her in particular for her help analysing the results gained from testing the Black Spot modules and the Point Grey Research camera initially used in this research.

I would like to thank Brent Price from ARANZ Scanning Ltd. for providing the opportunity to carry out this research. Brent has provided significant company resources and financial support. His drive to see the project develop has allowed me to purchase necessary equipment. Brent and his staff were responsible for designing the test jig used to collect the majority of the results in this project. Brent made a hardware modification to the Black Spot modules' image-data buses that saved a great deal of time.

I would also like to acknowledge the help of Shannon Carlson for his part in designing and building of the mechanical test jigs used in this project. These jigs included test beacons, a fiber glass enclosure for the camera modules, and the linear translation rails.

In the early stages of the project Oliver Ostojic from the University of Applied Sciences in Darmstadt, Germany worked with me. He helped write and maintain much of the early C# and MATLAB scripts for collecting data from the Point Grey Research camera.

This research was funded in part by a Technology Fellowship (TIF) from the Foundation for Research Science and Technology (FRST). Without this funding it would not have been possible to complete this project and it was greatly appreciated.

Finally, I would like to thank my friends and family for their support and for proof-reading many drafts of my thesis.

Contents

Abbreviations

The abbreviations used in this thesis are listed here for a reference.

3D	Three Dimensions/Dimensional
ASL	ARANZ Scanning LTD.
ASCII	American Standard Code for Information Interchange
CCD	Charge Coupled Device
CG	Computer Graphics
CLCC	Ceramic Leadless Chip Carrier
CMOS	Complimentary Metal Oxide Semiconductor
CTS	Clear To Send
BGA	Ball Grid Array
DMA	Direct Memory Access
DOF	Degrees of Freedom
DSP	Digital Signal Processor
FOV	Field of View
FPGA	Field Programmable Gate Array
FTDI	Future Technology Devices International LTD.
GCC	Gnu Compiler Collection
GDB	GNU project Debugger
GNU	GNUs not UNIX
GPS	Global Positioning System
GPL	GNU General Public Licence
IRL	Industrial Research Limited
ISO	International Standards Organisation
JTAG	Joint Test Action Group
LED	Light Emitting Diode
LEPD	Linear Effect Photo Diode
LVDS	Low Voltage Differential Signalling
OO	Object Oriented
PC	Personal Computer

PCB	Printed Circuit Board
PPI	Parallel Peripheral Interface
RISC	Reduced Instruction Set Computing
RMS	Root Mean Square
ROI	Region of Interest
RTOS	Real Time Operating System
SDRAM	Synchronous Dynamic Random Access Memory
SRAM	Static Random Access Memory
SNR	Singal-to-noise ratio
SPI	Serial Peripheral Interface
SPORT	Serial Port
TWI	Two Wire Interface
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
VGA	Video Graphics Array
VR	Virtual Reality

Chapter 1

Introduction and Background

Laser scanners are used in a diversity of fields to produce 3D models of objects. During the making of "The Lord of the Rings", the FastSCAN laser scanner was used to scan clay characters for Computer Graphics (CG) operators to animate [1]. By sweeping a laser beam produced by the scanner across the surface of an object, the 3D coordinates of surface points are determined and a model of that surface constructed. Laser scanners have been used in the medical [2, 3] and film industries [1, 4], and for reverse engineering [5], restoration [6], and the cataloging of artifacts [7, 8].

The FastSCAN uses a Fastrak [9] electromagnetic tracking system provided by Polhemus Ltd. to determine the position and orientation (the 'pose') of the scanner. The intersection between a line of laser light emitted from the scanner head and the surface of an object is recorded by one or two cameras attached to the scanner. Using these intersections and knowledge of the scanner pose, the surface points can be calculated precisely.

ARANZ Scanning Ltd. (ASL), a New Zealand company developed FastSCAN and produces this laser scanner in conjunction with Polhemus Ltd. Figure 1.1 shows the FastSCAN Cobra(TM) [10] that is one of two models of laser scanner offered by the companies. In this figure, the operator is shown holding the scanner. The laser line is emitted from the top of the scanner and the camera is below the operator's hand.

While accurate 3D models can be produced using this system, the Fastrak has limitations. Fluctuations in the magnetic field strength can lead to distortions in the models produced. Such fluctuations can be caused by ferromagnetic objects near the tracking system and interfering fields such as mains power fields [11]. This limits the materials that can be scanned accurately to non-ferrous objects. It would be of great benefit if a scanner could scan objects containing ferrous materials.

Figure 1.1 The FastSCAN Cobra(TM)is shown in this figure. It is capable of scanning the surfaces of objects and building intricate 3D models using a PC.

Another difficulty with the Fastrak system is the limited volume in which it operates. Using the standard transmitter, this is a sphere with a radius of approximately 76 cm, and tracking outside of this volume leads to lower quality tracking data [9]. Scanning of larger objects requires multiple scans to be 'spliced' together which is time consuming. These difficulties have motivated ASL to investigate different tracking technologies in the hope of increasing the tracking range and enabling their scanner to work with ferromagnetic objects.

It is believed that an optical tracking system could be used to augment the current system and lead to a hybrid design that may overcome the problems associated with magnetic tracking. Optical systems do not rely on magnetic properties to operate and therefore ferromagnetic objects do not affect these systems. The placement of the optical 'markers' (light sources) used in these systems dictate the extents of the 'tracking volume'. This is the 3D space that tracking occurs in. By adding more markers, tracking volumes may be extended relatively easily. However, using light sources introduces new problems such as reflections from shiny surfaces and interference from ambient light. Occlusion of optical markers can also be a problem. A hybrid system could overcome the problems inherent in both types of design.

The purpose of this work is to design and build a prototype six Degrees-of-Freedom (DOF) optical tracking system that can be used with FastSCAN. This system will enable the scanner's pose to be determined as it moves in 3D space. The Cartesian X, Y, and Z coordinates and the yaw, pitch, and roll comprise the six DOF.

A rigid array of cameras operating in the visible light spectrum detect fixed light sources around the periphery of a room. These cameras are attached to the laser scanner. By continuously calculating the cameras' positions relative to these fixed light sources, the

pose of the scanner can be estimated.

This work combines tracking theory, embedded software design, PC software design, and hardware design to produce a prototype that can be developed into a fully-fledged tracking system in the future.

While the aim is to build a prototype system, the emphasis has been on producing the "Black Spot" prototype camera modules that form the rigid group of cameras. Due to this focus, the thesis concentrates on reporting the design and performance of the camera modules rather than the entire system. In Section 1.1 existing tracking technologies are discussed. This is followed by an overview of available optical tracking systems in Section 1.2. Finally, a summary of the structure of this thesis is given in Section 1.3.

1.1 Review of tracking systems

Many methods exist for determining the pose of an object. Although the focus of this project is on an optical system, it is worth noting other tracking methods. Mechanical, acoustic, inertial, magnetic, and radio frequency trackers have all been implemented in the past, however, just like optical tracking, each method has its advantages and disadvantages [12]. Mechanical trackers typically have a mechanical linkage between the object being tracked and a fixed point. As the object moves, its position can be determined using measurements of transducers attached to the mechanical links. This method has the disadvantage that the pose of the object is limited by the physical constraints of the mechanical links. A commercial example of a mechanical tracker is the FaroArm by Faro Technologies [13].

Inertial trackers use gyroscopes and accelerometers as building blocks. These are available as microelectronic mechanical systems (MEMS) in integrated circuits. Inertial trackers have several advantages. They do not rely on any line of sight measurements as do optical trackers. They are not affected by magnetic fields or acoustic noise. They have a low latency and a high pose output rate. The big disadvantage with these systems is that the sensing components drift from their initial calibration. Given the small size of MEMS devices, inertial systems could be integrated into a super tracker that relies on optical, magnetic, and inertial tracking.

Acoustic tracking uses sound to determine the position of an object. Commercial systems are based on the time of flight of an ultrasonic sound wave. The performances of these systems are affected by temperature, humidity, and airflow. Typically, they, have slower update rates than inertial trackers. An example of a hybrid acoustic and inertial tracking system is the InterSense IS-900 [14]. That 6 DOF system uses ultrasonic range sensors to

compensate for drift in measurements from accelerometers and gyroscopes.

Electromagnetic trackers, such as the Polhemus Fastrak [9], use magnetic fields to determine pose. These systems use source and sensor units. The source units create AC or DC magnetic fields that are picked up by the sensor unit or units. Either the source unit or sensor unit (system dependent) contains three coils oriented so as to create or measure magnetic fields in the three orthogonal directions. As noted, these systems suffer from distortions due to ferromagnetic objects in the vicinity of the object being tracked. Unlike optical tracking systems, and to a lesser degree radio frequency tracking systems, magnetic trackers do not require the source and sensor to have a line of sight.

Radio frequency (and microwave) trackers use electromagnetic waves to enable tracking. Usually these systems use time of flight, however, this is more difficult to implement compared with acoustic trackers as the speed of light is many times faster than the speed of sound and precision electronics are required. Perhaps the most well known radio frequency tracking system is the Global Positioning System (GPS) run by the United States Department of Defence [15]. This system uses a constellation of between 24 and 32 Medium Earth Orbit satellites. Using signals from a number of satellites, approximate positional tracking (in comparison to other systems) can be performed. Radio frequency tracking systems suffer from multipath issues. This occurs when waves bounce off objects and cause multiple signals to be received. This adds to the difficulties of designing a radio frequency tracker. The author is not aware of any precision radio frequency trackers available on the market.

The last type of tracking system and the type chosen for this research is optical tracking. These systems use light sources (optical markers) and optical sensors. The position of the markers with respect to the sensors enables these systems to calculate pose. A summary of some of the available optical systems is given in Section 1.2.

1.2 Existing optical systems

A large amount of work has been performed in the area of optical tracking. Many optical tracking systems exist and these systems can be broken into two broad categories: inside-looking-out systems and outside-looking-in systems. An inside-looking-out system uses markers outside of the tracking volume. These are observed by the optical system mounted on the object whose pose is being estimated. An outside-looking-in system has the observation component of the system on the outside of the tracking volume and uses optical markers mounted on the object within the tracking volume. An example of an optical tracker that uses the inside-looking-out model is the 3rdTech HiBall tracker [16] and an example of an outside-looking-in tracker is the optical tracking system from Advanced

Realtime Tracking [17]. Both of these systems are briefly described below, followed by a description of a system under development by a New Zealand research institute, Industrial Research Limited (IRL) [18].

1.2.1 The HiBall tracker

The HiBall optical tracker [16] began life at the University of North Carolina at Chapel Hill in the early 1990s. The project uses arrays of ceiling mounted infra-red LEDs as markers. The tracking sensor is a small device that the designers have termed a HiBall. This sensor uses Linear Effect Photo Diodes (LEPD) to detect the positions of the LEDs relative to the HiBall. These LEPD devices produce a current proportional to the position of the light that falls on the diodes. The HiBall has a complex optical design that allows the number of LEPDs to be minimised while still providing a large Field of View (FOV). The system uses many hundreds of LEDs mounted precisely in ceiling panels. Using a Kalman Filter [19] based approach the tracker is able to accurately produce a pose estimate. The system can generate 2000 pose estimates per second with less than 0.5 mm and 0.02 degrees of position and orientation noise. This research aims to prototype a similar system to the HiBall tracker but it is intended to use cameras instead of LEPDs and to use fewer LEDs.

1.2.2 The Advanced Realtime Tracking system

An example of an outside-looking-in tracking system is produced by a German company named Advanced Realtime Tracking. They produce tracking hardware and software named ARTtrack and DTrack [20]. In contrast to the HiBall tracker this system uses specially designed cameras that emit flashes of infra-red light. These cameras surround the object being tracked. Light from the cameras bounces off reflective spheres attached to the object being tracked and is reflected back to the cameras. The orientation and position of the spheres are known with respect to a reference point and this information is exploited to allow the object's pose to be calculated. This system has the advantage that many objects can be tracked concurrently as reflective markers are cheap and can be attached easily. An indication of the performance of this system could not be found on the company website.

1.2.3 Industrial Research Limited's prototype system

Industrial Research Limited (IRL), a New Zealand research institution, has published papers detailing their research into an optical scanning system [21, 22]. This system uses infra-red LEDs as does the HiBall tracker but uses cameras to sense the position of the LEDs instead of using LEPDs. The cameras are mounted in a fixed orientation and position and observe the infra-red LEDs placed on the outside of the system. Using the images from the cameras, the system is able to determine the position and orientation of the cameras.

It is designed for high precision and for operating both indoors and outdoors in relatively large tracking volumes. The prototype IRL tracker uses PC based systems to do this. A disadvantage of this approach is that PCs are often larger than embedded systems that run dedicated firmware. However, an advantage of this approach is that algorithms can be designed and tested easily using a PC. It is believed that this system is designed with large scene surveys in mind rather than accurate object scanning. It appears that the accuracy is on the order of a few millimetres based on the figures in one of their papers [21].

1.2.4 Other optical tracking systems

There are other optical tracking systems on the market. A list of some of these systems is summarised in Table 1.1 along with their salient features. The systems in this list serve different purposes. For example, the Phoenix Technologies Inc. Visualeyez system provides 3 DOF for body motion capture. The OPTOTRAK PRO system from Northern Digital is designed for making discrete point measurements. Neither of these systems are suitable for use with FastSCAN. Of the systems listed in this table, only the HiBall tracker from 3rdTech would be suitable due to its high accuracy and 6 DOF output. Depending on the performance of the IRL prototype, this system could also be used with FastSCAN.

1.3 Thesis structure

The remainder of this thesis describes the design of the system and its performance. The system in this research uses an array of intelligent camera modules in fixed orientations relative to each other. They observe a number of LEDs placed on the perimeter of the tracking volume. An overview of the design is given in Chapter 2. A specification for the system is developed and the topology of the system described. The components and their roles are discussed. Following this, Chapter 3 analyses the design and describes the theory behind the design. In the following chapters the specifics of the system are described, beginning with the camera module firmware (Chapter 4), followed by the hardware (Chapter 5), the communications system (Chapter 6), and the central part of the design known as the hub (Chapter 7). The thesis finishes by presenting results in Chapter 8, focusing on the 2D performance of the camera modules and touching on the overall system performance. Finally Chapter 9 provides conclusions from the research and identifies areas for future work.

Name	Company	Description	Features	Website
ARTtrack/ DTrack [17]	Advanced Realtime Tracking GmbH	A tracking system using custom cameras that observe the positions of reflective spheres by using pulse of infra-red light.	Tracks multiple objects, 3 DOF or 6 DOF.	http://www.ar-tracking.de
HiBall Tracker [23]	3rdTech	Fast accurate tracker that was developed at the University of North Carolina at Chapel Hill. Uses photo diodes and LEDs to estimate pose.	Low latency, 2000 Hz update rate, Better than 0.2 mm positional precision and better than 0.01 degrees angular precision.	http://www.3rdtech.com/
IRL prototype	Industrial Research LTD.	Prototype system. Not currently in production.	Indoor and outdoor use.	http://www.irl.cri.nz/
Laser Bird [24]	Ascension	Designed for head and object tracking in medical and 3D visualisation applications.	Laser tracking technology.	http://www.5dt.com/
OPTOTRAK PRO [25]	Northern Digital	Optical laser tracker for making point coordinate measurements.	Laser tracking technology, 3DOF.	http://www.ndigital.com
Visualeyez [26]	Phoenix Technologies Inc.	A body motion capture system using LED markers.	0.5mm accuracy, 512 markers, 190 cubic metres tracking volume, 3DOF	http://www.ptiphoenix.com/

Table 1.1 A non-exhaustive list of optical tracking systems.

Chapter 2

Design Overview

This chapter provides an overview of the tracking system design. The major system components are described first followed by a discussion of design considerations. This discussion leads to the development of a specification for the system. The final section details the scope of the research.

2.1 System components

The design is for an inside-looking-out optical tracking system using camera modules. High intensity LEDs are used to provide feature points for tracking. In this research these are referred to as 'markers'. A collection of markers in a fixed configuration is known as a 'beacon'. The intention is to place multiple beacons around the periphery of the 'tracking volume'.

Like the IRL system [21] but unlike the HiBall tracker [16], cameras are used to determine the angles between the object being tracked and the markers around the tracking volume. Multiple camera modules with limited FOVs are combined to synthesise a larger angular FOV. This larger FOV allows the system to recover images from the camera modules corresponding to a range of directions. These images are analysed by the system to detect the presence of markers. The camera modules are to be fixed to a rigid structure known as the 'hub enclosure'. This is a mechanical housing that keeps the camera modules in fixed and known locations and orientations with respect to a point on the enclosure. Figure 2.1 shows a model of an enclosure design. The camera modules are embedded into the enclosure and their lens assemblies protrude outwards. In this model, the camera modules provide coverage of the entire hemisphere with 45 degree FOVs; this requires 17 camera modules. Using the 2D locations of the markers in each set of images retrieved from the camera modules, a pose estimation algorithm will calculate the pose of the hub enclosure

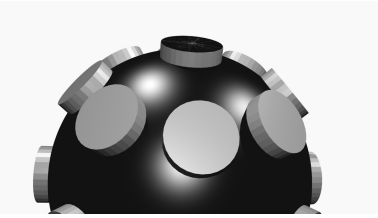


Figure 2.1 The hub enclosure for the optical tracker is to contain a number of camera modules that view the surroundings to provide, for example, a hemisphere coverage. In the figure the camera lenses are depicted protruding from the spherical hub.

with respect to a reference pose in the tracking volume.

Development of the system requires creation of hardware and software. The pose estimation algorithm and the beacon location algorithm require both a hardware platform to run on and a software development environment. Ideally the pose algorithm would be run on an embedded system in the hub enclosure, however, for ease of initial development this was implemented on a PC with the intention that this will be ported to an embedded system once a prototype had been created. This PC software was named the 'Soft Hub'.

It was decided that the camera modules would be small 'intelligent' embedded systems. They are designed to be modular system blocks, allowing them to be added or removed easily. They are responsible for locating the optical beacons in view and tracking their positions during motion. This information is passed back to the hub. As there are a number of camera modules in the system it is important that they are small so that they can fit into the hub enclosure.

A schematic of the system is shown in Figure 2.2. A number of optical beacons are shown placed inside the tracking volume. The camera modules observe these beacons and return the locations of the individual markers within these beacons to the hub. The pose is calculated using this system and is the final output on the right of the figure.

2.1.1 Beacons

Beacons consist of a number of markers arranged in a known configuration. As noted, these are the feature points that are tracked in this system and are fixed around the outside of the tracking volume. Green 510 nm 8 mm wide-angle LEDs [27] were chosen as markers in the system. This was motivated by the desire to reduce the number of LEDs when compared with systems such as the HiBall tracker [16]. It is believed that by using wide angle LEDs, i.e., LEDs that can be imaged from a large angle off their principal axis, the number of required LEDs can be reduced. A reduction in the number of LEDs leads to a cheaper system and increased ease of installation. A single green LED is shown in Figure

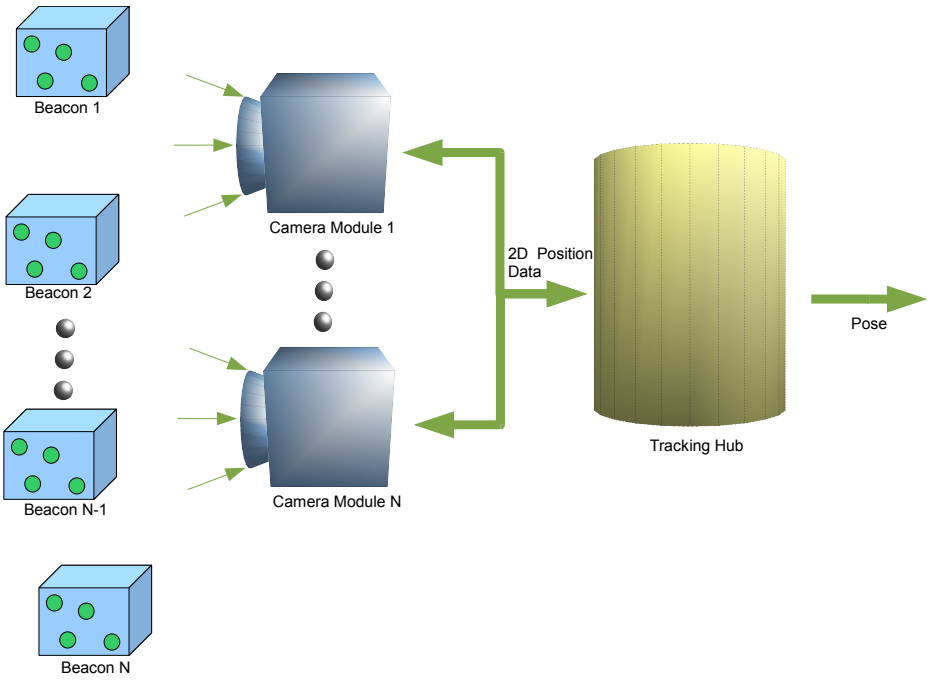


Figure 2.2 The tracking system comprises beacons, camera modules, and a hub. The beacons are in fixed locations around the periphery of the tracking volume and the array of camera modules observe them. The camera modules are mounted in a rigid frame named the 'hub enclosure'. Details of the beacons' positions are sent to the hub and these are processed to output an estimate of the 'hub enclosure's' pose.

2.3a and, as shown, is visible when viewed at up to 90 degrees from its principal axis.

The beacons are designed to be tethered and the LEDs to be driven using constant current sources to provide constant irradiances. It may be necessary for the LEDs to be modulated in the future and for the beacons to be battery powered. This would require a small embedded system to be used to control the beacons. This was investigated briefly but abandoned in favour of a simpler constant current source design.

The beacon used for testing consists of an array of 27 markers placed in a regular 9 by 3 pattern. Beacons comprising multiple markers are used in testing for three reasons. Firstly, they can be used to verify that a camera module can track multiple markers. Secondly, a beacon containing multiple markers can be used to calibrate for the distortions in a camera module's lens using existing calibration methods [28, 29]. Finally, using multiple markers per beacon reduces the cost of the system. The cost of producing a beacon that consists of one marker is comparable to the cost of producing one that contains multiple markers because both designs need a mechanical housing and power supply. The cost of a single LED is small compared with these other items. Therefore, it is beneficial from a cost point of view to produce a tracking system that uses a small number of beacons.

(a) (b)

Figure 2.3 (a) Green 8mm LEDs are used as feature points for tracking by the system. These are referred to as 'markers' throughout this research. (b) Markers are incorporated into beacons. These are groups of markers in fixed locations. A beacon comprising four markers is shown in this figure.

In most cases it is possible to uniquely estimate pose using only four markers [30]. If four markers are integrated into one beacon then pose estimates can be made when a single beacon is in view instead of requiring, in the single marker per beacon case, four beacons. Figure 2.3b shows an example of a beacon using four markers placed at the edge of the structure. Beacons such as this one were used during development and were constructed by embedding markers into a plastic housing as shown in the figure. A configuration using a number of these four-marker beacons placed around the tracking volume is preferred.

2.1.2 Camera modules

The camera modules are the eyes of the system. They calculate the positions of the markers they observe within a stream of image frames and pass this information back to the hub. From this information the hub can determine the angles to the markers. During camera motion, the relative motion of the markers' images must be tracked as they move within each image frame.

Each module contains a powerful processing unit capable of analysing a constant stream of images from a sensor placed behind a lens and filter assembly. The processing unit is linked to a communications system implemented in hardware and software. In this research, three camera module prototypes were built. However, it is intended that more of these modules may be used in the future to provide a greater combined FOV. Due to their function to detect bright spots (markers), and the Blackfin processing unit used in the camera modules, the name 'Black Spot' was coined for the camera modules. The camera module firmware was written using ISO C [31] as described in Chapter 4.

2.1.3 The hub

The hub is the brains of the system. It uses the locations of the markers from the camera modules along with a model of the surroundings to construct an estimate of its pose. The

hub is the concept that is implemented by the Soft Hub and the hub enclosure. The C# language [32] was chosen for development of the Soft Hub.

2.2 Design considerations

The design of the system is constrained by a number of parameters including size, weight, power consumption, and tracking performance. In a prototype design some of these constraints can be relaxed but they are still considered so that a final design may be created that does not require major changes from the prototype. It is important that the tracking system performs sufficiently well so that it can be used with the laser scanner. This section attempts to determine the constraints on the system. As the system is intended to augment or replace the existing tracker, it makes sense that it should perform as well or better than the Fastrak.

2.2.1 Operating assumptions

The system is intended to be operated indoors in a semi-controlled environment (for the prototype and foreseeable future). It is assumed that lighting can be controlled. The hub is intended to be run on a PC and communicate with the camera modules. Beacons are designed to be fixed to the walls and ceiling. The only motion between the hub and the beacons is assumed to be due to movement in the hub. As beacons are intended to be fixed to walls this seems a fair assumption. However, vibrations from, for example, air conditioning units could cause the beacons to move slightly. A marker is assumed to be a point source whose apparent position does not change with respect to the angle at which it is viewed. It is assumed that a hemispheric coverage by the modules will be sufficient to provide tracking. Finally, it is assumed that partial occlusions can be handled by the system. It is believed that with calibration, optimising the number of beacons, beacon placement, and optimisation of the number of camera modules and their placement the system specification can be met.

2.2.2 Size

The tracking system must be small enough such that it can be attached to the ASL scanner without compromising usability. There are two options for the system and these affect the size of the tracker. The first option is to build the physical enclosure that holds the camera modules into the scanner itself. For example, each face of the scanning head could have a camera mounted to it or embedded into the scanner head itself. This setup would have the advantage of reducing the overall size of the tracking system but would mean that the tracker could only be used with the scanner due to its integration into the scanner body. Integrating the camera modules into the scanner would also limit the orientations of the

camera modules to angles perpendicular to the scanner faces. Also, the construction of a scanner would be complicated by including camera modules into the structure.

The second option is to build a separate enclosure for the camera modules and attach this to the scanner. This would increase the overall size of the scanner and have a visual impact. The orientations of the modules would not be limited as in the first scenario. For the prototype constructed in this research the second option was chosen. However, in order to reduce the overall size for a commercial system, building the tracker into the scanner body would be a better option.

The maximum size of the tracker if it were built into an enclosure and attached to the scanner should be small enough so that the look and feel of the system is not compromised. The FastSCAN Scorpion(TM) [10] has a length of approximately 36 cm and a width of approximately 13 cm. By placing the hub enclosure on the top of the scanner and by keeping the enclosure diameter within the width of the scanner, the size of the scanner is not increased significantly. Based on these measurements, a sphere of diameter 10 cm would be a suitable basis for the tracker.

2.2.3 Weight

The system must be light enough for ease of use of the scanner. If the tracker is too heavy then this could limit how long an operator could comfortably use the scanner. Government agencies world-wide have guidelines on the maximum weight employees should lift. However, data could not be found on acceptable weights for sustained lifting such as could be encountered by an operator using the scanner. A maximum weight of 500 g for the tracker was chosen based on the weight of common 500 g mineral water bottles. These are of a weight that can easily be lifted using one arm (even in a state of dehydration). It is assumed that the weight of the scanner can be lifted easily by one of the operator's arms and the other arm could support the extra weight of the optical tracker. The scanner would therefore be held in both hands.

2.2.4 Operating temperature

The system is intended for use indoors and should work in conditions found in a typical office environment. It is assumed an office environment would not fall below 5 °C or rise higher than 35 °C.

2.2.5 Power consumption

The system is intended to be used in a tethered environment so power consumption is not a major issue from an energy conservation point of view. Therefore, power consumption is

not specified in this chapter. However, increased power consumption results in increased heat output. As many camera modules will be packed closely together, it is important to consider the heat transfer between a camera module and its surroundings based on a known power input. To estimate this, information about the camera module hardware design is required. A brief analysis is given in Chapter 5.

2.2.6 Humidity

Humidity should not affect the system unless the air condenses in which case it could form a layer of moisture on the optics that could degrade the system performance.

2.2.7 Tracking volume

The system must overcome the key limitations of an electromagnetic tracking system. These are the limited tracking volume and distortions due to metallic objects. The latter is overcome by the optical nature of the system but the former must be specified. The Fastrak manual [9] specifies a working radius of 76 cm using standard equipment and this corresponds to a working volume of approximately 1 m^3. The Black Spot modules are designed to be able to track the locations of markers as close as 0.5 m from a camera module and as far as 3 m. This suggests a tracking volume within a cube of dimensions 3 m $\times$ 3 m $\times$ 3 m, giving a volume of

$$V_{\text{track}} = 3^3 = 27\,\text{m}^3. \tag{2.1}$$

2.2.8 Tracking performance

Tracking performance relates to a number of parameters that affect the quality of the data returned by the tracker. These are the tracker resolution, accuracy, latency, and update rate.

Tracking resolution and accuracy

Tracker accuracy is the most important system parameter when used in conjunction with a handheld laser scanner. Accuracy describes how closely a pose estimate matches the real pose. As pose is a position and orientation, accuracy describes both positional accuracy (how close the estimated position is to the real position) and angular accuracy (how close the estimated orientation is to the real orientation).

Resolution is related to accuracy but corresponds to the number of distinct values the output pose can take. It can be thought of as the uncalibrated accuracy. It defines an upper limit on the accuracy a system could achieve. Calibration is not considered in this research so while an accuracy is defined, the goal is not to achieve this with the prototype design

but to produce a system capable of a high resolution.

The Fastrak resolution is defined in the manual [9] . The positional resolution decreases with range. The resolution at a distance of 1 cm is approximately 5 μm RMS. The angular resolution is quoted at 0.025° RMS. The goal is to achieve resolutions of at least 5 μm and 0.025°RMS over the entire tracking volume and allow future calibration to achieve the accuracy of the Fastrak or better. The Fastrak has a positional accuracy of 0.8 mm and 0.15° angular accuracy. In this research resolution is quoted in terms of standard deviations instead of root-mean-square measures (RMS) that the Fastrak manual uses. Both measures are interchangeable.

Update rate

The Polhemus tracker achieves an update rate of between 30 and 120 Hz depending on the number of receiver units in use. In a standard one-receiver configuration the rate is 120 Hz. This is the rate at which pose estimates are generated. The update rate affects the speed at which an object can be scanned. A greater update rate means a faster scanning speed can be used to produce the same quality scanned data. Initial component identification revealed that many image sensors do not support rates as high as 120 frames / s. It was determined that an update rate of 60 Hz was likely to be fast enough. This is justified in Chapter 3. If this number needs to be increased in the future there are options available to do this. These are firstly to use an image sensor that supports higher frame rates and secondly to synthetically increase the update rate.

The image sensor frame rate does not necessarily equate to the system update rate. There are two options to increase the update rate without changing the frame rate. As there will be a number of camera modules in use at a given time, the first option is to offset the clocks belonging to the modules' image sensors with respect to each other. By offsetting the update point for each module the update rate could effectively be increased by a factor equal to the number of camera modules present in the system. For example, for three modules the update rate could be lifted from 60 Hz to 180 Hz. The second method is similar to the first but involves calculating a pose estimate as soon as a marker location is updated by a camera module. For example, if one camera module is tracking 10 markers then this could potentially increase the update rate by 10 times. Care would need to be taken to ensure that the pose estimation algorithm worked correctly with marker positions that could come from different instants.

Tracking latency

Latency measures the time it takes for data at the input to the system to become available at the output of the system. This is the responsiveness of the system and affects the user

experience. This lag can be frustrating as anyone who has held a long distance phone call can verify. The Virtual Reality (VR) community use the term 'presence' to describe the feeling that a user is immersed in a virtual reality environment. An increase in latency is shown to reduce the sense of presence [33]. In this research a large latency means that real changes in pose are not seen at the tracker output and consequently the computer screen for some time. Although this application is not a virtual reality system it seems reasonable to assume that, like VR systems, a sense of presence is felt when using the laser scanner with a low latency. Watson et al. [34] note that previous studies aimed at finding a minimum latency required to maintain a sense of presence in VR environments have concluded with a range of values. The lowest reported by Watson et al. was from research by Wickens and Baker [35] that concluded that performance on a simulator can be impaired with lags as low as 50 ms. This value of 50 ms is chosen as the maximum allowable end-to-end latency for this system. The end-to-end latency for the entire system is the time measured from when laser light hits the surface of the object until when the corresponding data points are seen on the computer screen.

2.3 System specification

The design considerations in Section 5.3 lead to the system specification in Table 2.1. The most challenging specification to achieve is the accuracy. Calibration of the system is required to achieve the high accuracy specified and this is a significant task. It was decided during this research that calibration of the system was beyond the scope of the project. Another specification that could potentially be difficult to achieve is the latency of 50 ms. This is because at 60 frames/s, 16 ms is required by the camera modules for each data update leaving only 34 ms to send the data to the hub and produce a pose estimate. Parameters such as size, weight, temperature range, and humidity are rough estimates and may need to be revised. The primary focus of the research is on the design of the Black Spot camera modules and this is reflected in the remainder of this thesis. By developing a robust camera module, the system can be built on a strong foundation allowing the accuracy specification to be met in the future.

	Description	Value
Volume of operation	The volume in which the tracker can operate.	$3\,\mathrm{m} \times 3\,\mathrm{m} \times 3\,\mathrm{m}$
Positional accuracy	The positional accuracy expressed as one standard deviation of a sample of data points.	0.8 mm
Angular accuracy	The angular accuracy.	0.15 °
Positional resolution	The resolution of the position output.	$5\,\mu\mathrm{m}$
Angular resolution	The resolution of the orientation output.	0.025 °
Tracking latency	The latency of the system measured from a discrete movement by an operator until a new pose estimate is made.	50 ms
Update rate	The rate at which the system makes pose estimates.	60 Hz
Size	The maximum dimensions of the tracker. The prototype that was built is based around a fiber glass enclosure that is not spherical and accommodates only three modules.	0.1 m diameter sphere
Weight	The weight of the entire tracker.	500 g
Temperature range	A range of temperatures that the scanner can operate in at the accuracy specified.	5 - 35° C
Humidity	The percentage relative humidity.	Non-condensing

Table 2.1 The desired parameters of the tracking system are shown here. The prototype produced will aim to meet a subset of these parameters. This will not include the accuracy specification as system calibration is not considered.

Chapter 3

Theory and Design Analysis

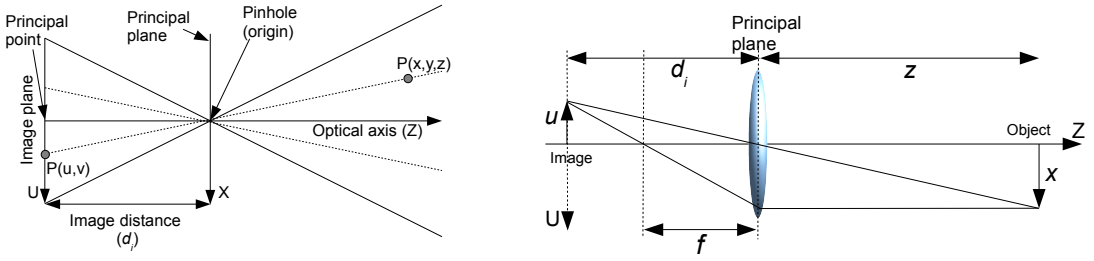
This chapter analyses the design of the optical tracking system described in Chapter 2. The major functions of the system are the tracking of markers within captured images, the calculation of the positions of the markers, and the estimation of the pose from this data. Firstly, supporting theory for this analysis is introduced in Section 3.1. Following this, the Black Spot design is analysed in Section 3.2. A brief review of the pose estimation literature is given in Section 3.3. Finally, a possible beacon configuration is described in Section 3.4.

3.1 Supporting theory

This section introduces quaternions, coordinate frames, and a basic camera model required by the hub's pose estimation algorithm. A coordinate frame describes the orientation and position of a 3D Cartesian coordinate system with respect to another coordinate system. The tracking system requires mathematical modeling of a camera and a coordinate frame to describe the pose of the camera. Quaternions are used to describe the rotation of a set of axes with respect to another. Firstly, the camera model is introduced. This is followed by quaternions and finally coordinate frames are introduced. It is assumed that the reader is familiar with basic vector operations described in many mathematical texts [36, 37].

3.1.1 Camera model

A pinhole camera model [38] is used by the software to form a projection that maps 3D scene coordinates to 2D camera coordinates. A pinhole camera has an infinitely small aperture (pinhole) and an image plane (Figure 3.1a). Light rays pass through this pinhole and form an image on the image plane. The mapping or perspective transform relates the coordinates $[x, y, z]$ of a point in 3D space to the 2D coordinates $[u, v]$ of the point on the



(a) (b)

Figure 3.1 (a) The pinhole camera model. All light passes through the infinitely small pinhole forming an image on the image plane. (b) The thin lens approximation. Light passes through the convex lens forming an image at a distance d_i from the principal point.

camera's image plane. This relationship is

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{xd_i}{z} \\ \frac{yd_i}{z} \end{bmatrix}, \tag{3.1}$$

where d_i is the distance between the image and the principal plane. In real world cameras this pinhole is replaced with a lens (Figure 3.1b). A convex lens can be modeled using a thin lens approximation.

Thin lens equation

The thin lens equation is

$$\frac{1}{f} = \frac{1}{d_i} + \frac{1}{z} \tag{3.2}$$

and relates the focal length (f), the image distance (d_i), and the object distance (z). The focal length is a property of the lens and is specified by the lens manufacturer. If the distance to the object is much greater than the focal length then Equation 3.2 can be approximated by

$$d_i \approx f \tag{3.3}$$

which states that the image plane is at a distance f from the centre of the lens aperture. Due to the magnitudes of the numbers involved in this system (a focal length of 6 mm was used with a minimum marker distance of 0.5 m) this approximation is used and Equation 3.1 can be written as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{xf}{z} \\ \frac{yf}{z} \end{bmatrix}. \tag{3.4}$$

Lens aperture

The lens aperture determines how much light enters the lens. This is normally specified with a quantity named 'f-number' and sometimes written $f/\#$. This number is a ratio of focal length to the clear aperture 'ϕ' (the effective lens diameter) as shown by

$$\mathrm{f-number} = \frac{f}{\phi}. \tag{3.5}$$

A large f-number corresponds to a small aperture and therefore lets less light enter the lens. A small f-number gives a large aperture and lets more light into the lens. A detailed discussion on optics is given in Born and Wolf [39].

3.1.2 Quaternions

Quaternions are used in this research to describe the orientation of coordinate frames. Quaternions are the outcome of research by W.R. Hamilton in the 18th century [40]. They are hyper-complex numbers and form a 4 dimensional vector space over the real numbers. The four components of a quaternion q can be written

$$q = w + x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k}. \tag{3.6}$$

Here w represents a scalar and x, y, z are often combined into a vector $\boldsymbol{v}$. The terms $\boldsymbol{i}$, $\boldsymbol{j}$, and $\boldsymbol{k}$ are similar to the imaginary term $\boldsymbol{i}$ in a complex number and are orthogonal with respect to each other.

Quaternions can be used to elegantly describe rotations. Other representations such as Euler Angles, rotation matrices, and the angle and axis representation exist [41–43]. However, quaternions do not suffer from a degenerative problem known as *gimbal lock* [41–43]. This can affect the composition of rotations using the Euler Angle and rotation matrix representation of rotations resulting in a loss of one degree of freedom at some orientations.

A quaternion of unit length represents a rotation. The length or norm of a quaternion is similar to the standard vector definition, i.e.,

$$|q| = \sqrt{w^2 + x^2 + y^2 + z^2}. \tag{3.7}$$

Quaternion multiplication, unlike multiplication of the real numbers, does not commute, i.e., for two quaternions q_1, q_2, $q_1 q_2 \neq q_2 q_1$. To determine the product of the two quaternions the components of each quaternion are multiplied term-by-term. The multiplication

identities for multiplying $\boldsymbol{i}$, $\boldsymbol{j}$, and $\boldsymbol{k}$ are

$$\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1 \tag{3.8}$$

and from these identities, the remaining products such as $\boldsymbol{ij} = \boldsymbol{k}$ can be derived.

Using Equation 3.6 and Equation 3.8, the quaternion product of q_1, q_2 is

$$\begin{aligned} q_1 q_2 &= (w_1 + x_1\boldsymbol{i} + y_1\boldsymbol{j} + z_1\boldsymbol{k})(w_2 + x_2\boldsymbol{i} + y_2\boldsymbol{j} + z_2\boldsymbol{k}) \\ &= w_1 w_2 + \\ &\quad (w_1 x_2 + x_1 w_2 - x_1 x_2 + y_1 z_2 - z_1 y_2)\boldsymbol{i} + \\ &\quad (w_1 y_2 + y_1 w_2 - y_1 y_2 - x_1 z_2 + z_1 x_2)\boldsymbol{j} + \\ &\quad (w_1 z_2 + z_1 w_2 - z_1 z_2 - x_1 y_2 - y_1 x_2)\boldsymbol{k} \end{aligned} \tag{3.9}$$

The addition of quaternions q_1, q_2 is defined in the same manner as vectors, i.e.,

$$q_1 + q_2 = w_1 + w_2 + (x_1 + x_2)\boldsymbol{i} + (y_1 + y_2)\boldsymbol{j} + (z_1 + z_2)\boldsymbol{k}. \tag{3.10}$$

Quaternion rotations can be constructed in terms of an axis and a rotation angle. A quaternion q can be defined using a rotation around an axis (l, m, n) by an angle θ using

$$q = \cos\frac{\theta}{2} + l\sin\frac{\theta}{2}\boldsymbol{i} + m\sin\frac{\theta}{2}\boldsymbol{j} + n\sin\frac{\theta}{2}\boldsymbol{k}. \tag{3.11}$$

For Equation 3.11 to hold, the axis (l, m, n) must be normalised to length 1.

Similarly an axis and angle can be determined from the components of a quaternion q using

$$\begin{aligned} w &= \cos\frac{\theta}{2} \\ \sqrt{x^2 + y^2 + z^2} &= \sin\frac{\theta}{2} \end{aligned} \tag{3.12}$$

that was derived from the definition of a quaternion. This leads to an angle and axis of

$$\begin{aligned} \theta &= 2\cos^{-1} w, \\ l &= \frac{x}{\sqrt{1 - w^2}}, \\ m &= \frac{y}{\sqrt{1 - w^2}}, \\ n &= \frac{z}{\sqrt{1 - w^2}}. \end{aligned} \tag{3.13}$$

Given two 3D vectors $\boldsymbol{v}$ and $\boldsymbol{v'}$ there is a quaternion q_r that describes the rotation between $\boldsymbol{v}$ and $\boldsymbol{v'}$ given by

$$q_{v'} = q_r q_v q_r^{-1}. \tag{3.14}$$

Here q_v is a quaternion which has a vector part $\boldsymbol{v}$ (the scalar part can take any value), $q_{v'}$ is the quaternion with vector part $\boldsymbol{v'}$, and q_r^{-1} is the inverse of the quaternion q_r. For quaternion rotations, finding the inverse of q is simplified as it is equal to the conjugate of q, i.e., $q^{-1} = q^*$and the conjugate of a quaternion is defined as

$$q^* = w - x\boldsymbol{i} - y\boldsymbol{j} - z\boldsymbol{k}. \tag{3.15}$$

The components of a quaternion are expressed in short hand in this thesis using the notation $q = (w, x, y, z)$ where w, x, y, z are the components of the quaternion. The notation $q = (w, \boldsymbol{v})$ is also used. Here w represents the scalar part and $\boldsymbol{v}$ represents the vector part. Quaternion rotations are used extensively throughout the implementation of the Soft Hub (see Chapter 7) to describe orientations. They are also used in the construction of coordinate frames described in Section 3.1.3.

3.1.3 Coordinate frames

A coordinate frame is the term used in this work to describe the orientation and position of the standard orthogonal X, Y, and Z axes in 3D with respect to another three axes. A coordinate frame is implemented here using a vector and quaternion pair. The vector represents the translation of the coordinate frame's origin relative to a 'parent' frame. Likewise, the quaternion represents the orientation of the axes in terms of these two frames. Figure 3.2 shows the axes of two coordinate frames. A point $\boldsymbol{P}$ in the parent frame (X, Y, Z axes) is transformed to a point $\boldsymbol{P'}$ in the X′, Y′, Z′ frame by

$$\boldsymbol{P'} = \boldsymbol{T} + q_R \boldsymbol{P} q_R^{-1}, \tag{3.16}$$

where $\boldsymbol{T}$ represents the translation vector that translates the origin from the parent frame to the new frame and q_R is a quaternion that describes the rotation that transforms a vector in the parent frame to a vector in the new frame.

3.2 Black Spot module analysis

The Black Spot modules perform 2D tracking of markers. The locations of the markers are sent to the hub after each frame. There are three steps required to achieve this: segmentation, position calculation, and tracking. Before discussing each of these, the concept of a region of interest (ROI) is introduced.

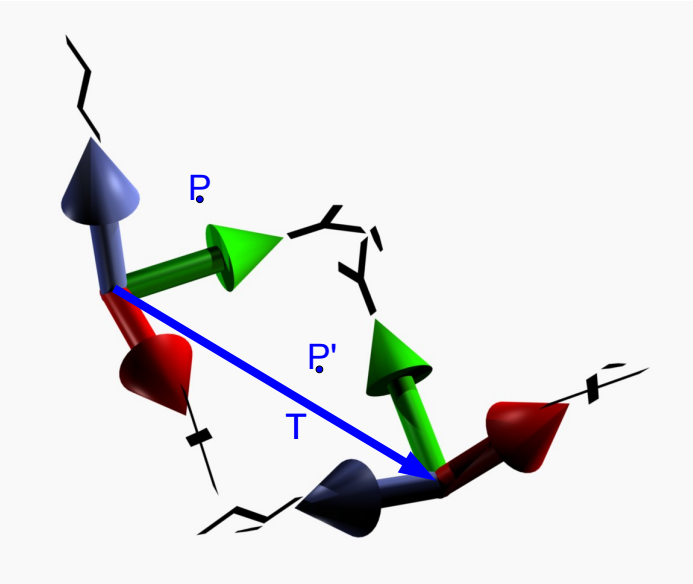


Figure 3.2 A coordinate frame is defined as a translation and rotation with respect to a parent frame. Two sets of axes are shown here. The X', Y', Z' axes are defined with respect to the X, Y, Z axes (the parent frame). A translation from the parent frame origin to the X', Y', Z' origin is given by $\boldsymbol{T}$.

3.2.1 Regions of interest

A region of interest (ROI) is a rectangular window with sides parallel to the image frame that defines an important area of an image. All image processing operations in the Black Spot firmware operate on ROIs. The firmware is responsible for identifying these regions, creating them, and removing them. By using ROIs the number of pixels that must be analysed per frame is significantly reduced. A ROI contains the image of a marker. As the marker's location within the Black Spot's FOV changes, the position of the ROI is updated to follow the marker's image. Figure 3.3 shows the composition of a ROI. The marker is shown in the middle of the region. An area of background pixels surround the marker. By making the ROI larger than the marker, the firmware can track changes in location. This process is described in Section 3.2.5. During the remainder of this thesis the term 'marker' will be used to describe both a marker and a marker's image.

3.2.2 Marker segmentation

In order to track the positions of markers in a stream of images, the markers must be identified and segmented from the background. A trivial method to segment an object from its surroundings is by background subtraction. This method uses a reference background image frame and this is subtracted from the given frame. The result of this subtraction is an image that contains the difference between the foreground and the background. Provided

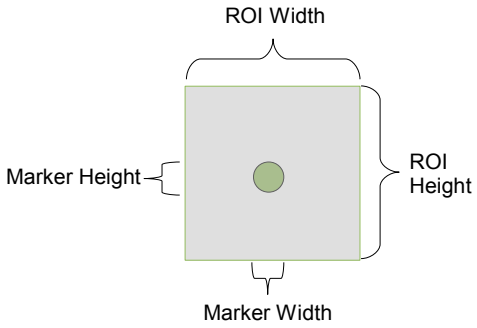


Figure 3.3 A Region of Interest (ROI) is a rectangular window that surrounds a marker.

that the background remains constant, foreground objects can be emphasised using this approach. The method cannot be used in this system as the Black Spot modules themselves are moving. Therefore, there cannot be any stationary reference image.

Another method of segmentation is template matching [44–46]. A template image that is to be detected is compared with an image frame. The template image must be shifted across the entire captured image and the similarity between the template object and the image data in the template position determined. If the similarity is high enough then the algorithm determines that the object is present at this shifted position. Similarity can be defined in a number of ways including the cross-correlation between the image and the background it covers, as well as the sum of the squared differences between the pixels from the template and image [44–46].

Template matching relies on the image template remaining constant. However, as the camera moves in relation to the markers, the images of the markers will change in size. This can be accounted by using a number of templates of markers at different distances but this requires more memory. Template matching is also computationally intensive (due to the template being shifted many times) and has not been considered here.

The approach taken to segment the markers is computationally cheap and can run in real-time on an embedded system. Under the limited conditions that the system is designed to work in, the markers are assumed to be the brightest objects in the image. The algorithm looks for bright pixels in each image. If a bright pixel is located, then a ROI is constructed. The pixels above a certain threshold are assumed to be bright image pixels. Therefore, segmentation is a combination of using a ROI and this threshold. It is assumed that the marker will be contained within this ROI. This is typically the case but, depending on the position of the bright pixel, the marker may be only partially contained by the ROI. Also, the bright pixel may correspond to a non-marker object, such as a reflection or image noise. Despite these potential issues this method works well in the office setting used for testing. Further explanation is given in Chapter 4.

3.2.3 Marker position calculation

Shortis et al. 1994 [47] investigate a number of techniques for precisely locating a target. In their paper they analyse the performance of centroids, Gaussian shape fitting, and ellipse fitting using a simulated environment. Their simulations show that the centroid performs well for precisely locating a target under a variety of conditions. This research uses the centroid of a marker to estimate position within the ROI. Shortis et al. 1995 [48] investigated centroid and ellipse fitting algorithms using real data from video sequences and concluded that, in general, centroid algorithms are robust and insensitive to threshold level changes. The intensity weighted centroid is used in this research to precisely locate a marker within a ROI.

The calculation of the centroid of an image [47] is analogous to finding the centre of mass of an object where the intensity of an image is equivalent to mass. The centroid of an image is the intensity weighted sum of the points that make up that image and is therefore a vector-valued function. The calculations of the X, and Y components of the centroid with respect to the ROI position are

$$C_{\mathrm{ROI}} = \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \frac{\sum_{j=1}^{m}\sum_{i=1}^{n} I(i,j)i}{\sum_{j=1}^{m}\sum_{i=1}^{n} I(i,j)} \\ \frac{\sum_{j=1}^{m}\sum_{i=1}^{n} I(i,j)j}{\sum_{j=1}^{m}\sum_{i=1}^{n} I(i,j)} \end{bmatrix}, \tag{3.17}$$

where $I(i,j)$ denotes the intensity of the pixel at position $[i,j]$ within the ROI of dimensions n by m. A binary centroid calculation is also possible with $I(i,j)$ set to 1 for all marker pixels. However, Shortis et al. [47,48] show that this has lower performance than the intensity weighted centroid.

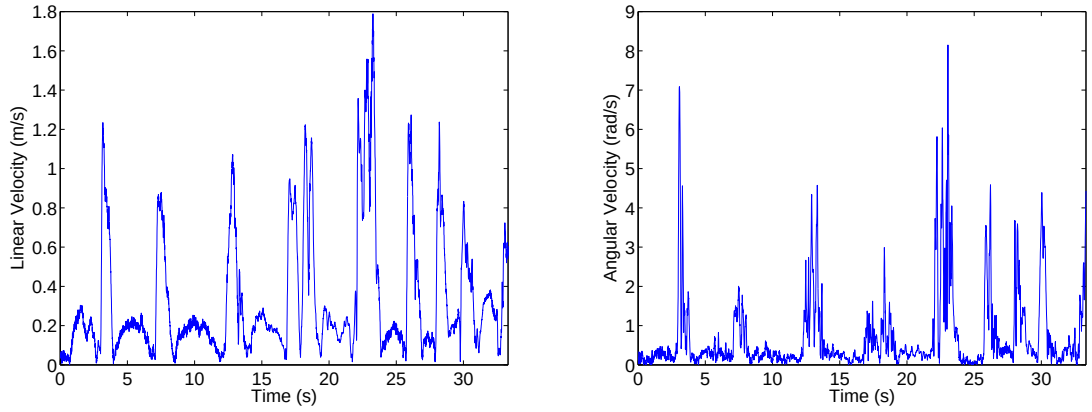
After calculating the location of a marker within its ROI, the location of the marker within the image can be calculated by

$$P_{\mathrm{Marker}} = P_{\mathrm{ROI}} + C_{\mathrm{ROI}}, \tag{3.18}$$

where P_{ROI} is the position of one of the corners of the ROI within the image. The corner chosen is the one with the lowest X and Y coordinates.

3.2.4 Tracking speed calculations

The use of camera modules to calculate the positions of markers introduces a complication which is not an issue with the Fastrak. An electromagnetic tracker can determine the pose of a target regardless of the motion that the target is undergoing. This is not the case for the optical tracker proposed in this thesis due to the method used to track markers. As noted, a ROI is used to surround a marker image. When a marker moves between two consecutive



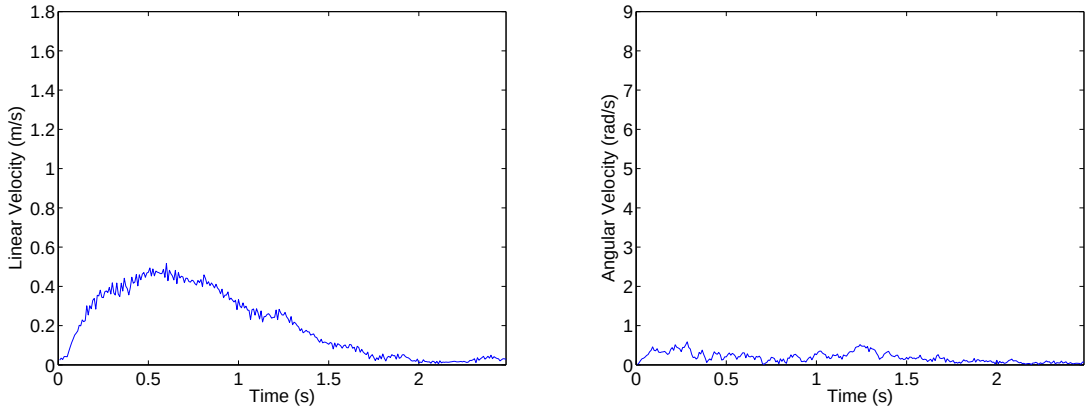
(a) (b)

Figure 3.4 The recorded velocities during a complete scan have spikes due to the operator re-aligning the scanner after a sweep. This can be seen in (a) the linear velocity and (b) the angular velocity.

frames, the ROI must follow it so that the image remains contained within the ROI. This section investigates how the scanner is typically used and the types of motion it undergoes during scans. Using this information, the required tracking speed can be estimated.

The system should be able to track motion at a rate that is high enough to enable the system to be used with the laser scanner. To gauge how the scanner is used, data were gathered for a typical scanning scenario. In this experiment, an experienced operator scanned on object and the motion of the scanner throughout the experiment was collected using a Fastrak unit. This data were analysed and the maximum angular and linear velocities and accelerations determined.

Figure 3.4a shows the linear velocity and Figure 3.4b shows the angular velocity during a scan. The scan consists of periods of low velocity with spikes of much higher velocity. This is apparent in both the linear velocity and angular velocity graphs. This can be explained by the way the operator uses the scanner. Generally an object is scanned in sweeps. The operator sweeps the scanner over a portion of the object at a fairly low speed allowing the system to capture the features of the object at a high resolution. At the end of a sweep it is typical for the operator to flick the scanner back to the top of the object before beginning another sweep. This can be seen in the periods of much higher velocity. Although it would be preferable to track the scanner throughout an entire scan, it is acceptable if the quality of the tracking is reduced substantially after the operator finishes a sweep and returns to the top of the object to re-commence scanning. As long as the system can return to a high quality tracking mode in a timely fashion, this period of reduced accuracy will not hinder the scanning procedure.



(a) (b)

Figure 3.5 (a) The linear velocity during a sweep. (b) The angular velocity during a sweep. These velocities do not contain the spikes that appear during an entire scan.

Assuming the spikes in velocity between sweeps can be ignored, the maximum velocities during a typical sweep may be analysed. Figure 3.5a shows the linear velocity of the scanner during a single sweep. The sweep starts off slowly and rises to a maximum velocity of approximately 0.52 m/s before dropping away again. The angular velocity shows a different picture (Figure 3.5b) and remains low throughout the sweep. Intuitively this seems reasonable since the operator keeps the scanner at a fairly constant angle to the surface being scanned. In a broad sense, the surface is likely to be flat during a sweep and therefore the angle of the scanner remains approximately constant. As the angular velocity is low (reaching a maximum of 0.59 rad/s) it is not analysed further. The linear velocity is analysed to determine the maximum tracking speed.

The maximum sweep velocity in m/s can be transformed into a velocity on the image plane in pixels/frame. These units will be the native units used by the firmware. For linear translations, parallax causes objects closer to a camera to appear to move faster than objects further away from the camera. The minimum working distance is 0.5 m so the maximum image velocity is calculated using this figure. It is a function of the velocity of the scanner, the FOV of the camera, the frame rate of the camera, and the distance between the markers and the camera. The velocity v' in pixels/frame is given by

$$v' = \frac{vfT_r}{zk}, \tag{3.19}$$

where v is the velocity in m/s, z is the distance between the camera origin (the pinhole) and the marker, k is the length of a pixel at the image sensor in metres, T_r is the frame period, and f is the focal length of the camera lens. This equation is derived using the perspective transform in Equation 3.4 and is illustrated in Figure 3.6. For example, using a focal length of 6 mm, a distance of 0.5 m between the camera and marker, a frame period of $1/60$ s, and

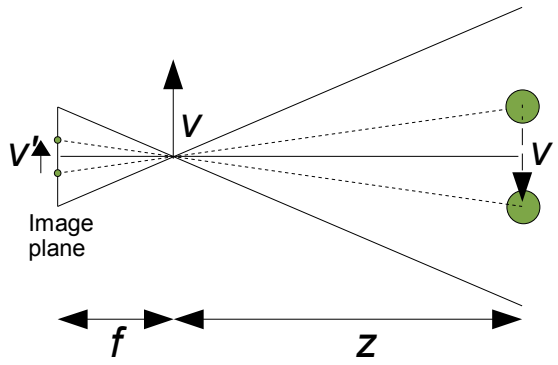


Figure 3.6 A translation of the camera and its corresponding effect on an image. The two green circles represent a marker in two different frames. For simplicity the marker is shown moving with respect to the camera, however in reality, the opposite is true as shown by the arrow at the origin in the diagram. By determining the maximum velocity during a sweep the maximum tracking speed required of the Black Spot firmware can be determined.

an approximate increase of 50% in the maximum sweep velocity[1] ($0.52 + 50\% \approx 0.8\,\mathrm{m/s}$), the equivalent image velocity is

$$v' = \frac{0.8 \times 6 \times 10^{-3}}{60 \times 0.5 \times 6 \times 10^{-6}} \approx 27\,\mathrm{pixels/frame}. \tag{3.20}$$

Using a similar method the linear acceleration was analysed. The maximum linear acceleration was approximately 9.1 $\mathrm{m/s^2}$ and this is approximately 14 $\mathrm{m/s^2}$ when increased by 50% leading to a value of

$$a' = \frac{afT_r^2}{zk} = \frac{14 \times 6 \times 10^{-3}}{60^2 \times 0.5 \times 6 \times 10^{-6}} = 8\,\mathrm{pixels/frame^2}. \tag{3.21}$$

These values are used in the design of the firmware as described in Chapter 4.

3.2.5 Marker tracking

To track the movement of an image from one frame to the next the marker's ROI must follow the marker. This is achieved by centring the ROI around the centroid of the marker after each frame (Figure 3.7). The translation of a marker across the camera's image plane can be estimated using the distance in pixels between the centroid of the ROI in the previous frame and the centroid in the current frame. The system's ability to track the movement of a marker is related to the magnitude of this image translation per frame and the size of the ROI. There is a trade-off between maximum image tracking speed that can be tracked

[1] 50% is used as a *safety* margin to account for the possibility that the scanned data represents a slower than average scan.

and ROI size. A larger ROI leads to a greater speed that may be tracked but large ROIs are undesirable as they require a greater number of pixels to be analysed per frame. If the ROI size is too small then the marker cannot be tracked as the marker could move outside the ROI within one frame period.

The required tracking speed affects the size of the ROIs used. The arrow in Figure 3.7 illustrates marker movement. For the marker to continue to be tracked, the magnitude of this movement must be small enough so that the marker remains within the ROI. This leads to a formula that describes the relationship between marker movement and minimum ROI Size, i.e.,

$$\text{ROI Size} \geqslant 2s_{\max} + D. \tag{3.22}$$

Here ROI Size refers to one of the dimensions of the ROI. Over one frame the maximum marker movement is equal to the maximum marker velocity $v'_{\max}$. This gives

$$\text{ROI Size} \geqslant 2v'_{\max} + D, \tag{3.23}$$

where D is the marker diameter. The marker diameter is a function of distance between the camera's origin and the marker. The maximum ROI size can be calculated by substituting D for the marker diameter when viewed at the closest tracking distance of 0.5 m. Using the green 8 mm diameter LEDs and optics described in Chapter 5 and using a pinhole camera model described in Section 3.1.1 this diameter is

$$D_{\max} = 16\,\text{pixels}. \tag{3.24}$$

Therefore, the ROI size required to track the maximum speed of 27 pixels / frame at 0.5 m is

$$\text{Max. ROI Size} \geqslant 2 \times 27 + 16 = 70\,\text{pixels}. \tag{3.25}$$

This corresponds to a ROI of 70 by 70 pixels. The maximum ROI size can be reduced to 32 by 32 pixels using an algorithm in Chapter 4 that compensates for the velocity of the marker and alters Equation 3.23 to depend on the maximum acceleration a'_{max}.

3.3 Pose estimation

The second major function of the system is the estimation of pose from the 2D data provided by the camera modules. There is a vast body of literature covering techniques to solve this type of problem. Solutions were first published by the photogrammetric community and rediscovered by the computer vision community. Consequently, there are a number of names that are used to describe the class of problem. Relevant areas in the literature have names such as multiview geometry [49], structure from motion [50], bundle

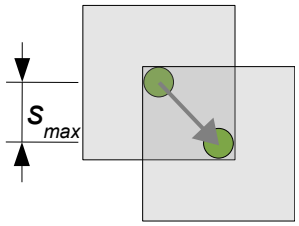


Figure 3.7 A marker is tracked by centring the ROI containing the image around the centroid after each frame. If a marker image moves outside of the bounds of the ROI then the marker will no longer be tracked. s_{max} is the maximum movement in the marker position that can be tracked using a ROI of this size.

adjustment [51], camera calibration [52], pose estimation [53–55], space resection [56], and the perspective-n-point problem [56].

Multiview geometry is a broad term describing problems in geometric reconstruction using multiple camera views. Structure from motion (also known as structure and motion) covers problems in refining 3D structure given a number of 2D image sequences. Bundle adjustment is part of this refining process and the estimate of camera pose is one of the outputs of a bundle adjustment algorithm. Bundle adjustment is described in more detail in Section 3.3.1. Camera calibration often produces the extrinsic parameters of a camera (the camera's pose) as a by-product of calculating the intrinsic parameters required for calibration. Pose estimation, space resection, and the perspective-n-point problem are all terms that describe the problem of determining the pose of a calibrated camera given known reference points [56]. Recent development in these areas includes work by Snavely, Seitz, and Szeliski [57, 58] on reconstructing 3D scenes using multiple images available from public internet galleries such as Flickr [59].

Within the pose estimation literature there are methods based on analytic solutions. These are solutions to what is termed the perspective-3-point [56, 60, 61], perspective-4-point [30, 56], and perspective-n-point problems [56]. Quan and Lan [56] note that the pose estimation problem can be solved with only three known markers but that four markers are required to produce a unique solution.

There are also numerical methods for solving this pose problem based around minimising some sort of cost function. Usually these methods are presented as a minimisation of an error residual in a least mean squares sense [53–55, 62]. If there are outliers in the data then convergence can be compromised. Methods are available to pre-process the data and remove outliers [53, 63]. Each of these pose estimation methods assumes that the model points are known to infinite precision. It is unrealistic to assume this accuracy but the positions of markers can be surveyed to reasonable accuracy and this can then be used to provide a basic pose estimate.

In this thesis a pose estimation method is proposed that assumes model points are known to high precision. This method uses quaternions to describe the orientation of camera modules. An iterative method is used to refine the estimate of the modules' orientations. From the orientation, the position can be determined. In future research bundle adjustment [51] could produce much better results by removing the assumption that the model points are known precisely.

3.3.1 Bundle adjustment

Bundle adjustment is a broader topic compared with pose estimation. As well as being used to solve the same problem, i.e., that of estimating the pose of an object, it can also be used to refine the positions of model points. Depending on the bundle adjustment algorithm used, the intrinsic camera parameters can also be refined as part of the optimisation process. Triggs [51] gives a detailed overview of bundle adjustment methods. Triggs defines bundle adjustment as follows *"Bundle adjustment is the problem of refining a visual reconstruction to produce jointly optimal 3D structure and viewing parameter (camera pose and/or calibration) estimates."*.

Triggs continues that bundle adjustment is a large geometric parameter optimisation problem. The optimisation seeks to minimise an error function that in some way describes the difference between measured data and data predicted using a system model. Instead of the model parameters remaining constant as in the case of the pose estimation algorithms above, they are allowed to vary during the bundle adjustment.

In this research the measured data are the 2D centroids of the marker images. The model data include the pose of the hub enclosure, the coordinate frames of the cameras with respect to the hub enclosure, the focal lengths, and principal points of the cameras (intrinsic parameters), and the positions of the markers in 3D coordinates.

Implementing a bundle adjustment algorithm is beyond the scope of this research but the author believes that bundle adjustment is necessary to achieve the accuracy in the system specification. Ideas for how to approach this are given in the future work section of this thesis.

3.4 Beacon configuration analysis

A brief analysis of a possible configuration of the system is given in this section. To decide on the number of beacons and camera modules required to provide tracking throughout a $3\,\mathrm{m} \times 3\,\mathrm{m} \times 3\,\mathrm{m}$ volume, the effect a single beacon has on the tracking volume must be considered. Consider a single beacon fixed in space. For this calculation the beacon is

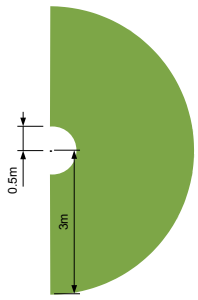


Figure 3.8 The maximum tracking volume (shaded) surrounding one beacon (dot) is determined by the minimum and maximum distances that the camera modules can observe markers, assuming the hub has a FOV that covers all angles.

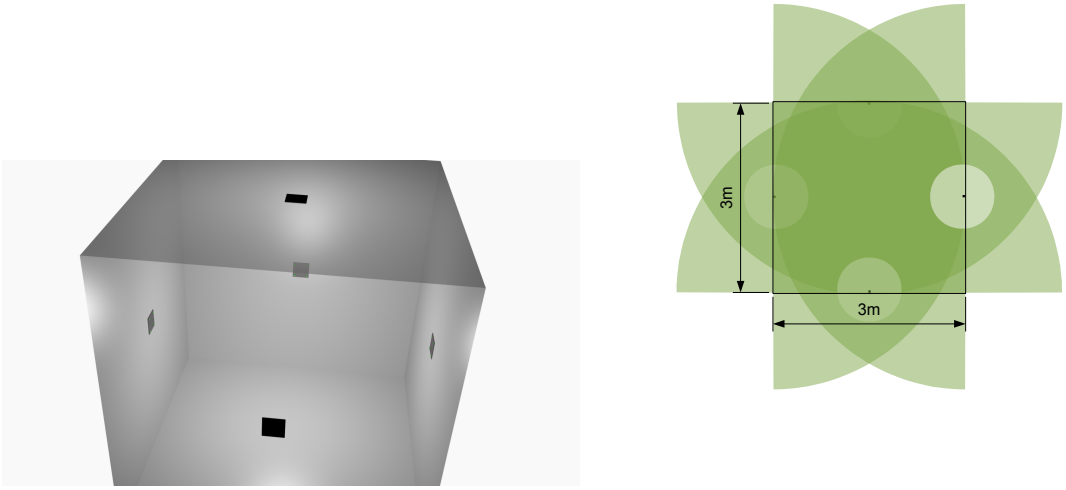
a point in space, the markers can be imaged at up to ± 90 degrees from their optical axes, and a pose estimate can be generated if the beacon is visible at any of these angles. Assume that the hub has a FOV in all directions. The tracking volume under these assumptions is a hemisphere with a radius of 3 m minus a hemisphere with a radius of 0.5 m (a 2D diagram is shown in Figure 3.8). This corresponds to a tracking volume of $V = \frac{2}{3}\pi 3^3 - \frac{2}{3}\pi 0.5^3 = 56\,\mathrm{m}^3$.

A possible design based around using beacons consisting of 4 markers is considered. Figure 3.9a shows a cube of dimensions $3\,\mathrm{m} \times 3\,\mathrm{m} \times 3\,\mathrm{m}$ that matches the tracking volume given in the system specification. Beacons are placed in the centre of each face of the cube except for the bottom face. This gives a total of 5 beacons and 20 markers. The tracking volumes arising from the side beacons are shown overlaid on top of each other in Figure 3.9b using the assumptions above. This illustration shows better tracking coverage in the centre of the cube with poorer coverage in the corners of the cube.

In reality the hub has a limited FOV and multiple beacons are needed to allow pose estimation to occur in all orientations throughout the tracking volume. It is difficult to estimate the number of beacons required to do this as it is a function of the FOV of the hub, the placement of the beacons, the number of markers required to produce an accurate pose, and the angles at which the markers may be imaged effectively. Using a number of camera modules it is believed that the configuration shown in Figure 3.9a can be used to provide tracking within most of the cube in many orientations, however, further development is required to test this.

3.5 Summary

The design behind the main functions of the system is analysed. Marker images are located by searching for bright pixels in an image frame. Regions of Interest (ROIs) are placed around the bright pixels and surround the marker images. The centroid of the pix-



(a) (b)

Figure 3.9 (a) The 3 m $\times$ 3 m $\times$ 3 m tracking volume containing 5 beacons with a total of 20 markers is shown. The beacons are placed on each wall and the roof. There are not any beacons on the floor. (b) The tracking volumes arising from each of the side beacons are shown overlaid on top of each other. The outline of the tracking cube is shown by the solid line.

els above a threshold in each ROI is calculated. The centroid coordinates are passed to the Soft Hub for use in pose estimation. A possible system configuration uses five beacons, each containing four markers, placed in the centre of five of the faces of a 3 m $\times$ 3 m $\times$ 3 m cube. There would not be any beacons placed on the floor.

By analysing the motion that a laser scanner undergoes during a typical scan, the maximum velocities and accelerations can be determined. A maximum linear velocity of 27 pixels/frame was calculated using data from a scan. This corresponds to a ROI size of 70 by 70 pixels.

Chapter 4

Black Spot Firmware

This chapter describes the design of the firmware for the Black Spot camera modules. As discussed in Chapter 2, the Black Spot modules provide 2D tracking of LED markers for the overall system. The goals of the firmware design are summarised in Section 4.1 then the algorithm design is described in Section 4.2. This section includes a discussion of a number of algorithms that could be implemented in the future to improve the firmware (Section 4.2.5). Following this the development environment is described in Section 4.3 and finally the implementation details are briefly discussed in Section 4.4.

4.1 Design goals

There were a number of design goals for the Black Spot firmware, namely:

- to track multiple markers concurrently in 2D[1],
- to track markers at the full frame rate available to the firmware (60 frames/s),
- to maintain tracking of markers while undergoing motion that would be encountered during typical use of the FastScan system (as described in Chapter 3),
- to work in a memory limited environment typical of an embedded system,
- to allow the firmware to be developed independently from the hardware,
- to write the firmware so that it can be tested easily,
- to make the firmware modular so that new algorithms can be added in the future.

[1]The beacon used for testing contained 27 markers so this number is used throughout the thesis.

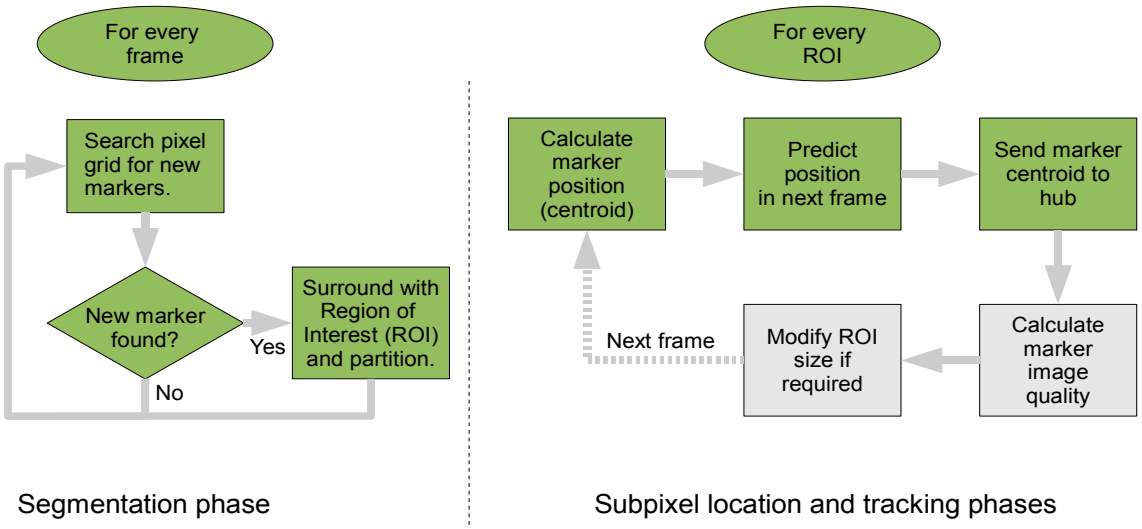


Figure 4.1 The firmware continually executes the steps in this diagram. These can be broken into a segmentation phase, a subpixel location phase, and a tracking phase. These phases involve finding marker images, creating ROIs, calculating marker image centroids and tracking ROI movement. The two boxes in gray represent algorithms that have not been implemented but should be implemented in the future.

The first three goals have implications for processing power and require the design of computationally light-weight algorithms. Routines were written to minimise the number of pixels from each frame that are used for computation. For example, an algorithm to predict marker motion based on previous positions is used and this allows fewer pixels to be analysed. In addition to minimising the number of pixels used, the memory footprint of the algorithms was also designed to be low to satisfy the fourth goal.

The last three goals relate to the development process. Ideas from the Object Oriented (OO) community were used to achieve these goals.

4.2 Algorithm design

The design of the algorithms is described in this section. A broad overview of the approach taken to tracking is described first followed by more detailed algorithm analysis. The approach to tracking was described in Chapter 3 and is reiterated here with a focus on implementation.

Figure 4.1 shows flow diagrams that identify the main parts of the firmware. Tracking is performed in three stages. The first is a segmentation phase that roughly determines the location of the markers being tracked. The second is a more accurate subpixel location phase that determines the markers' uncalibrated coordinates with high precision. This is followed by the tracking phase. These phases are repeated for each image frame. The

tracking phase includes two steps that are as yet unimplemented. These are the ROI size modification and the centroid quality calculation.

The segmentation phase is based around using ROIs. Every frame is searched for new markers. When a marker is found, a new ROI is created. The ROI is positioned so that it contains the marker image. Using the ROI and a signal intensity threshold[2], the marker image is crudely segmented from the background.

The motivation for using ROIs is that they greatly reduce the computation required. To illustrate this, suppose ROIs were not used. Every pixel of each frame would need to be analysed to determine whether it belonged to a marker. Further analysis would then be required to determine which marker the pixel belonged to. A brief calculation of required computing power in instructions per second shows this to be undesirable on a small embedded system. A centroid operation was written in C and compiled using GNU GCC for the Blackfin BF533 DSP [64]. By counting the instructions in the inner loop a value of approximately 25 instructions per pixel was calculated. While this figure will vary depending on the target processor and factors such as compiler optimisation, it probably represents a lower bound. Suppose that to determine whether a pixel belongs to a marker and if so which marker it belongs to requires at least the same number of instructions per pixel as the centroid calculation. For the wide VGA frame of 752×480 pixels used in this research (see Chapter 5), this results in a requirement of approximately

$$N_{\mathrm{FRAME}} = 752 \times 480 \times 25 \times 60 = 541\,\mathrm{M\,instructions/s\,(Mips)} \tag{4.1}$$

at 60 frames/s (fps). Again, it should be stressed that this is an absolute minimum.

Using ROIs, the subpixel phase involves the calculation of the centroid of the ROI. The ROI size is much smaller than the frame size so it is possible to implement efficiently on a small embedded system. Suppose there are 27 ROIs of size 32 x 32 and that each pixel requires 25 instructions to process. This results in

$$N_{\mathrm{ROI}} = 25 \times 32 \times 32 \times 27 \times 60 = 41.5\,\mathrm{Mips} \tag{4.2}$$

at 60 fps and is, therefore, a fraction of the computation required to analyse each pixel of every frame. These figures are rough estimates but, regardless of the actual number of operations required per pixel, using ROIs will result in fewer operations per frame purely because there are fewer pixels to analyse.

The tracking phase consists of moving the ROIs so that they surround the marker in the next frame. Using the position in the current frame and the previous frame, the position of

[2]This is the threshold above which pixels are assumed to belong to a marker image.

the marker in the next frame can be estimated. The ROI is centred on the new estimated position for the next frame. The marker centroid is sent to the hub during the tracking phase.

4.2.1 Initial marker detection

To enable markers to be tracked using ROIs, the approximate location of each marker needs to be determined so that it can be segmented from the background by the creation of a ROI. The algorithm designed for this initial detection looks for bright pixels in a captured image. If a pixel is above a specified threshold then it is flagged as belonging to a marker. A ROI is centred around this pixel and the centroid of the region is calculated after the next frame is captured. The threshold is a hard-coded constant. For the shutter period of 300 μs used during testing, an intensity value of 20 works well.

To reduce computational requirements, the algorithm is implemented using a grid. Only pixels on the corners of this grid are tested to determine whether they are bright pixels. For an 8 by 8 pixel grid, this reduces the number of pixels that must be checked by a factor of 64 and for a 16 by 16 pixel grid this is a factor of 256 times fewer pixels. This is a significant reduction in computation. During testing the grid was set to 16 pixels wide.

After each frame the grid is repositioned. This is an attempt to ensure all markers are located when the camera is stationary. For example, if a marker falls between the pixels on the grid then it will not be detected while the camera is motionless. If the grid is moved so that after a number of frames every pixel has been sampled then at some point the marker that was initially invisible will be discovered. For the 16 pixels wide grid it will take at the very most 256 frames before every pixel has been checked for a new marker. Therefore, at 60 fps, the worst case is that it will take $(16 \times 16)/60 = 4.3\,\text{s}$ to locate a marker assuming that it only covers 1 pixel. Based on the smallest marker image size at 3 m of approximately 2.5 by 2.5 pixels the grid can be moved by 2 pixels each time and still locate a marker of this size resulting in a further reduction in time of a factor of 4. This gives a figure of $(16 \times 16)/(2 \times 2 \times 60) = 1.1\,\text{s}$.

Figure 4.2 shows a screen-shot of the algorithm running on a PC. ROIs are shown as boxes with the centroid number inside the box. The grid is shown as the array of regularly spaced rectangles in the figure (these are enlarged for clarity). There are two further markers in the lower right hand corner, both of which coincide with grid pixels. The detection algorithm will detect these markers and ROIs will be created for them. The screen shot comes from a video stream that was created using a Point Grey Research FireflyMV [65] camera. This camera contains the same image sensor used by the Black Spot and was used to create test videos for the firmware before the Black Spot hardware was developed.

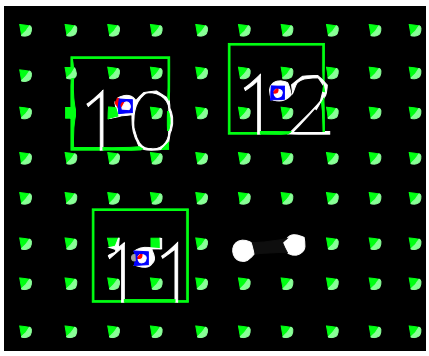


Figure 4.2 : A ROI is created when a bright pixel is detected at a grid point. The grid is moved every frame to allow stationary markers to be found. The numbers in the boxes identify the ROI.

(a) (b)

Figure 4.3 (a) A ROI can contain multiple markers. This can occur when a ROI is created around a group of markers that are far away from the camera. (b) Partitioning a ROI can solve this problem.

4.2.2 Multiple marker detection

During the segmentation phase, the initial marker detection algorithm can create ROIs that contain multiple markers. This occurs when the size of the ROI is large compared with the size of the markers' images (Figure 4.3a) and can happen when a beacon is far from the camera. In this case the default ROI size will cause multiple markers to be encompassed by one ROI. An algorithm that can split the ROI into smaller regions was developed for this situation.

Pixel connectivity and projection approaches were considered. A pixel connectivity approach is suggested by Shortis et al. 1994 [47]. Every pixel in a ROI is traversed to find groups of connected pixels that are above a threshold. Each group forms a ROI. This method was discarded as it requires each pixel to be tested. Compared with the projection approach described below it is likely to be computationally expensive.

The projection approach is to look for lines that can be drawn between the markers in the ROI. Since a ROI in this software is defined using vertical and horizontal lines only, a method of partitioning can be developed using only vertical and horizontal lines. Figure 4.3b shows an example of partitioning a ROI containing three markers using two vertical lines.

(a) (b)

Figure 4.4 (a)When an imaginary light is shone across the ROI shadows are cast by markers onto a line below the ROI. (b)The original ROI has been partitioned into three smaller ROIs.

(a) (b)

Figure 4.5 (a)This ROI requires two passes to partition correctly. As shown, after one pass the left most ROI still contains two markers. (b) Partitioning of the ROI is complete after two passes.

By looking at Figure 4.3b, it should be obvious that the lines partition the ROI into three smaller ROIs containing one marker each. Turning this intuition into code is more difficult. How are these lines calculated? First, consider another example. The images of the markers can be thought of as solid objects through which light cannot pass. Now suppose a light is shone across the ROI. The markers cast shadows on an imaginary line that runs parallel to the bottom of the ROI. Figure 4.4a depicts this. The projection of the shadows onto this line gives the details of how to partition the ROI. The mid-points between the shadows defines the positions of the lines. Figure 4.4b shows ROIs created by partitioning the original ROI using the midpoints of the shadows.

A difficult example is shown in Figure 4.5a. In this example the projection in one direction is not enough to create ROIs containing only one marker. The first vertical projection creates ROIs containing, from left to right, two, one, and two markers. To separate these regions, each of these new ROIs is then projected in the horizontal direction resulting in Figure 4.5b. Only vertical and horizontal directions are used for projection as this simplifies the creation of the ROIs by allowing only vertically and horizontally aligned ROIs to be created.

This algorithm was implemented and is partially optimised. The algorithm looks for the

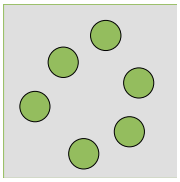


Figure 4.6 : A shape that will cause the partitioning algorithm to fail.

first marker pixel that would cast a 'shadow' in the direction of projection and once found ceases to look further as any further pixels found would not change the result, i.e., that there is a shadow.

There are cases that could cause the algorithm to fail. Consider the example in Figure 4.6. In this example the projections in both horizontal and vertical directions create continuous 'shadows', however, the ROI contains distinct markers. Beacons can be designed so that these shapes do not occur (or occur infrequently). Also, as partitioning only occurs in the segmentation phase, once a ROI is created it will not need to be partitioned again while it is being tracked.

4.2.3 ROI size adaptation algorithm

As the camera module moves, markers will appear to grow and shrink due to the changing distance between the camera and the markers. The ROIs that surround these markers must also grow and shrink so that each marker is surrounded entirely by its ROI and so ROIs do not overlap. The latter could happen if the camera moves away from a beacon. Figure 4.7a shows ROIs surrounding two markers. As the distance from the camera to the markers increases, an image pixel covers a greater physical distance on the marker. Therefore, the ROIs will cover a greater area around the markers and could overlap with an adjacent ROI as shown in Figure 4.7b. If the overlap is great enough to include part of the adjacent marker the centroid will be biased and could affect the pose estimate generated by the system.

The diameter of the marker within the ROI can be measured to determine whether the size of the ROI needs to be increased or decreased. Calculating the diameter of the marker can be achieved by using the formula for the area of a circle, i.e., $A = \pi r^2$. Criteria are needed to determine whether a marker image is too large compared with the size of its ROI or too small. A ROI is too small if it means that motion that could be encountered would cause the marker image to be lost, i.e., leave the ROI. Equations for the size of a ROI were developed in Chapter 3. This algorithm uses Equation 3.23 and a tolerance value `T` in pixels. Algorithm 1 illustrates the algorithm operation. In this code `S_ACTUAL` is the actual

(a) (b)

Figure 4.7 With a fixed ROI size, ROIs can overlap when the camera moves away from a beacon. This is because the number of pixels/m decreases as the distance from the camera increases. (a) shows two ROIs that do not overlap. (b) shows the same ROIs overlapping after the camera has moved away from the markers.

Algorithm 1 Pseudo code for a proposed algorithm to update ROI size is shown.

```
FOR EACH FRAME
   FOR EACH ROI
      LET S_REQUIRED = 2 * Vmax + D
      IF S_ACTUAL < S_REQUIRED THEN
         S_ACTUAL = S_REQUIRED + T/2
      ENDIF
      IF S_ACTUAL > S_REQUIRED + T THEN
         S_ACTUAL = S_REQUIRED + T/2
      ENDIF
   END LOOP
END LOOP
```

ROI size, `S_REQUIRED` is the required ROI size given by Equation 3.23. The tolerance `T` stops the ROI size changing rapidly. The algorithm ensures that the ROI remains between the size given by Equation 3.23 and this size plus the tolerance value. A small value for `T` of perhaps 5 pixels is suggested.

At the time of writing the algorithm for automatically adjusting the ROI size has not been implemented.

4.2.4 Marker position prediction

A method to improve the relationship between the ROI size required for a given tracking speed involves predicting the position of the marker. If the position of the marker in the next frame can be estimated, larger marker speeds can be tracked with smaller ROI sizes. Consider a marker moving at a constant velocity across the camera's image plane measured in pixels/frame. Since this velocity is constant, the position of the marker in the next frame can be predicted based on this velocity and using simple kinematic equations.

A linear prediction algorithm was devised to track the marker positions. Using the marker position at frame n and frame $n-1$, the average velocity of the marker between the two

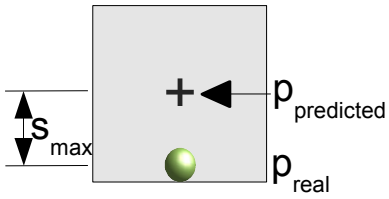


Figure 4.8 Using the linear prediction scheme, the ROI is positioned around the predicted point, $p_{\text{predicted}}$. The maximum deviation of the marker from the predicted point is shown and this is referred to as $s_{\max}$.

frames can be calculated by

$$\boldsymbol{v_n} = \boldsymbol{p_n} - \boldsymbol{p_{n-1}}, \tag{4.3}$$

where $\boldsymbol{v_n}$ is the velocity in pixels / frame and $\boldsymbol{p_n}$, $\boldsymbol{p_{n-1}}$ are the pixel positions at time n and $n-1$ respectively. The average acceleration can also be calculated by

$$\boldsymbol{a_n} = \boldsymbol{v_n} - \boldsymbol{v_{n-1}}. \tag{4.4}$$

Assuming that there are not any higher order terms then using kinematic equations the marker position in future frames can be calculated by

$$\boldsymbol{p_{n+N}} = N^2\boldsymbol{a_n} + N\boldsymbol{v_n} + \boldsymbol{p_n}, \tag{4.5}$$

where N is the number of frames in the future to predict. In the implementation of this algorithm N is always set to one as there is no need to predict the position of the marker image further than one frame into the future. This reduces Equation 4.5 to

$$\boldsymbol{p_{n+1}} = \boldsymbol{a_n} + \boldsymbol{v_n} + \boldsymbol{p_n}. \tag{4.6}$$

The algorithm is designed to predict the position in the next frame assuming constant velocity so takes the form

$$\boldsymbol{p_{\text{predicted}}} = \boldsymbol{v_n} + \boldsymbol{p_n}. \tag{4.7}$$

The ROI is centred around $\boldsymbol{p_{\text{predicted}}}$ after each frame. Figure 4.8 shows the maximum deviation from the predicted position ($s_{\max}$) that can be tolerated for a given ROI size. This is described in terms of the minimum ROI size required by

$$\text{ROI Size} \geqslant 2s_{\max} + D. \tag{4.8}$$

Using Figure 4.8 and Equation 4.6, $\boldsymbol{p_{\mathbf{real}}}$ is

$$\boldsymbol{p_{\mathbf{real}}} = \boldsymbol{a_n} + \boldsymbol{v_n} + \boldsymbol{p_n}. \tag{4.9}$$

In this case

$$s_{\max} = \left|\boldsymbol{p_{\mathbf{real}}} - \boldsymbol{p_{\mathbf{predicted}}}\right|. \tag{4.10}$$

Substituting Equation 4.9 and Equation 4.7 into Equation 4.10 gives

$$s_{\max} = \left|\boldsymbol{a_n}\right|. \tag{4.11}$$

In Figure 4.8, it is the maximum marker acceleration $\boldsymbol{a_{\mathbf{max}}}$ that defines $s_{\max}$. Equating the magnitude of this acceleration, $a'_{\max}$, to that observed in Chapter3 gives

$$s_{\max} = \left|\boldsymbol{a_{\mathbf{max}}}\right| = a'_{\max}. \tag{4.12}$$

Substituting $a'_{\max}$for $s_{\max}$ in Equation 4.8 gives

$$\text{ROI Size} \geqslant 2a'_{\max} + D, \tag{4.13}$$

which states that, using this prediction scheme, the minimum ROI size required to track the typical scanner motion found in Chapter 3 is proportional to the magnitude of the maximum acceleration. This was determined to be approximately 8 pixels / frame^2 meaning that the maximum ROI size required for a diameter of 16 pixels (Equation 3.24) is 32×32.

This prediction algorithm is a limiting case of a discrete Kalman Filter [19, 66]. The system state is the position $\boldsymbol{p_n}$ measured by the centroid calculation. The state prediction step is Equation 4.7. In this case, the centroid calculation is used directly to give the state estimate.

4.2.5 Future improvements

There are other algorithms that could improve the performance of the Black Spot module. They are proposed for future implementation.

Marker discriminator

The first algorithm is a marker discriminator algorithm. Ideally the module should be able to distinguish between a real marker and noise sources such as bright lights and reflections. Also, markers should be able to be ignored if they are too close or too far away to be useful. The design of this algorithm has not been investigated but the characteristics of marker images may be able to be used to determine whether a region of bright pixels is a marker or an artefact. A possible, although computationally intensive, approach would be to use

template matching [44–46] to match a marker image against a potential marker. This would be less intensive than template matching during segmentation because template matching would only be performed on each ROI rather than the entire frame. By calculating the size of the marker image, the approximate distance to the marker can be calculated and this could be used to decide whether the marker is too close or too far away to be used.

Duplicate ROI detection

An algorithm to detect whether identified ROIs are duplicates would be useful. In some cases it has been observed that rapid movement causes two ROIs to merge. This is movement greater than the maximum sweep velocity found in Chapter 3. This can be caused by a ROI containing part of an adjacent marker due to rapid movement. The centroid algorithm is then biased by the partially visible marker. The outcome is that the ROI slides on top of the other ROI and duplicates it. Periodically comparing the coordinates of known ROIs could provide a way to eliminate this problem.

Marker quality

Another useful algorithm would be one that assesses the quality of the markers based on measured properties of their images. Quality, in this context, is defined in terms of the variation in centroid positions while the Black Spot module is stationary. A high quality marker is one with a low centroid variation. Knowing the marker quality would aid 3D pose estimation since poor markers could be ignored or given a lower weighting. The quality metric could be a scalar equal to the weighted sum of a number of measured parameters. For example, these parameters could be the maximum pixel intensity, the extent of image saturation, and the size of the marker.

The peak pixel intensity is a useful measure as it has been shown that the location precision is approximately doubled for a doubling in peak intensity [67]. If the image is saturated then the maximum intensity cannot be represented by the number system. Shortis et al. 1994 [47] note that in their simulations, the location accuracy of the centroid calculation decreases with an increase in saturation of the image. The marker image size also has an affect on location precision. Clarke et al. 1993 [67] note that little improvement in precision is obtained by increasing the marker size above 5 pixels in diameter. Also, the position of the marker image in the ROI may be related to the quality of the centroid. For example, if the marker is not completely contained by the ROI then the centroid will not be comparable with the centroid of the full marker.

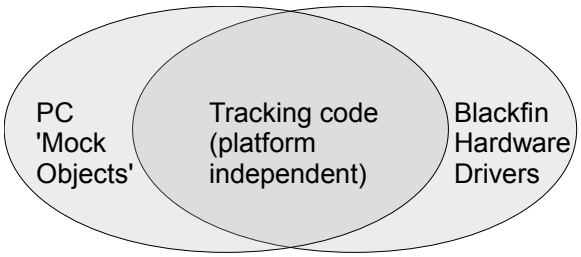


Figure 4.9 The firmware is designed to be cross-compiled for PC or the Blackfin processor. This is achieved by implementing hardware dependent code twice; once for the Blackfin and once for the PC. This allows development to begin before the hardware is complete and the firmware to be partially tested using a PC.

4.3 Development environment

The firmware is written in ISO/IEC C99 [31]. It was designed to be compiled either for a PC, using Microsoft Visual Studio, or the Blackfin camera module using GNU GCC and the open source Eclipse environment. This cross-compilation approach enables new algorithms to be developed and tested quickly using a PC. Development on a PC is preferred as it allows the full range of PC tools to be used as well as the use of resources that are not available on the embedded system. PC software can be easier to test for two reasons. First, testing can be easier to carry out due to the greater number of testing tools available. Second, hardware dependent modules can be substituted for *mock objects* (Figure 4.9). These mock objects mimic the behavior of the hardware but provide more control because, for example, unusual timing scenarios can be simulated and tested. Also, using mock objects allows testing to occur before the hardware platform is available [68]. For example, a number of test videos were used to test algorithms by substituting the data from the image sensor with the test video data. This was carried out before the hardware was built using the OpenCV computer vision library [69,70].

4.3.1 Debugging

Debugging is carried out using a PC wherever possible, however, the Black Spot hardware drivers must be debugged on the module. Debugging the firmware running on the Black Spot requires PC software and a special hardware debugging interface (Figure 4.10). The Eclipse IDE interfaces with a Blackfin specific version of the GNU Debugger (GDB) client running on the PC. This software allows Eclipse to control the execution of the firmware by setting breakpoints and stepping through the code line by line. The GDB client connects to a GDB proxy using a TCP/IP network connection. This allows the the firmware to be debugged from a remote machine if required. The GDB proxy software interprets the GDB debugging commands and translates them into commands that are sent to the debug module on the Blackfin using a IEEE 1149.1 [71] compatible interface board. This is commonly known as a Joint Test Action Group (JTAG) interface board. The GDB proxy communicates

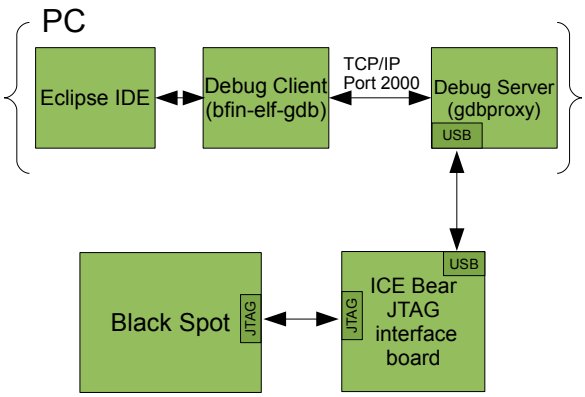


Figure 4.10 The firmware is tested on Black Spot modules using the ICE Bear hardware debugger connected via USB to a PC.

with the JTAG board via the Universal Serial Bus (USB). The JTAG board chosen was the ICE Bear from the Swiss company Section5 [72].

4.4 Implementation

This section briefly describes the firmware implementation. Figure 4.11 shows an UML class diagram of the core firmware functionality. As objects are not supported in ISO C this diagram is at best an approximation of the code and should be used to provide an overview of the code structure only. Some concrete classes have been omitted from the diagram for clarity leaving only the interface they implement. The diagram is split into two parts. These are the hardware dependent parts (on the left) and the classes that can be compiled on both a PC and the Blackfin DSP (on the right). The diagram is best described from left to right as this relates to the flow of data beginning at the image sensor in the Blackfin case or test data in the PC case and ending in the communications stream that is sent to the hub. The left part of the diagram contains hardware specific classes. These classes interface with the underlying hardware and therefore require implementations on both the PC and Blackfin. These classes include the image sensor drivers. The equivalent PC classes are the interface to the OpenCV library and test videos. The image sensor driver is described further in Section 4.4.1.

Starting from the first class that can be compiled on both PC and Blackfin, a frame strip is defined as a portion of a captured image frame. It has the width of the image frame and its height is a factor of the image frame's height. A class that implements the `IFrame-StripProvider` interface generates frame strips corresponding to portions of a captured frame. For example, with the 752 x 480 wide VGA frame used in this project, strips of height 24 pixels are used and this results in 20 strips per frame. These are consumed by a `ROIProvider` object using an event driven mechanism. The `IFrameStripProvider`

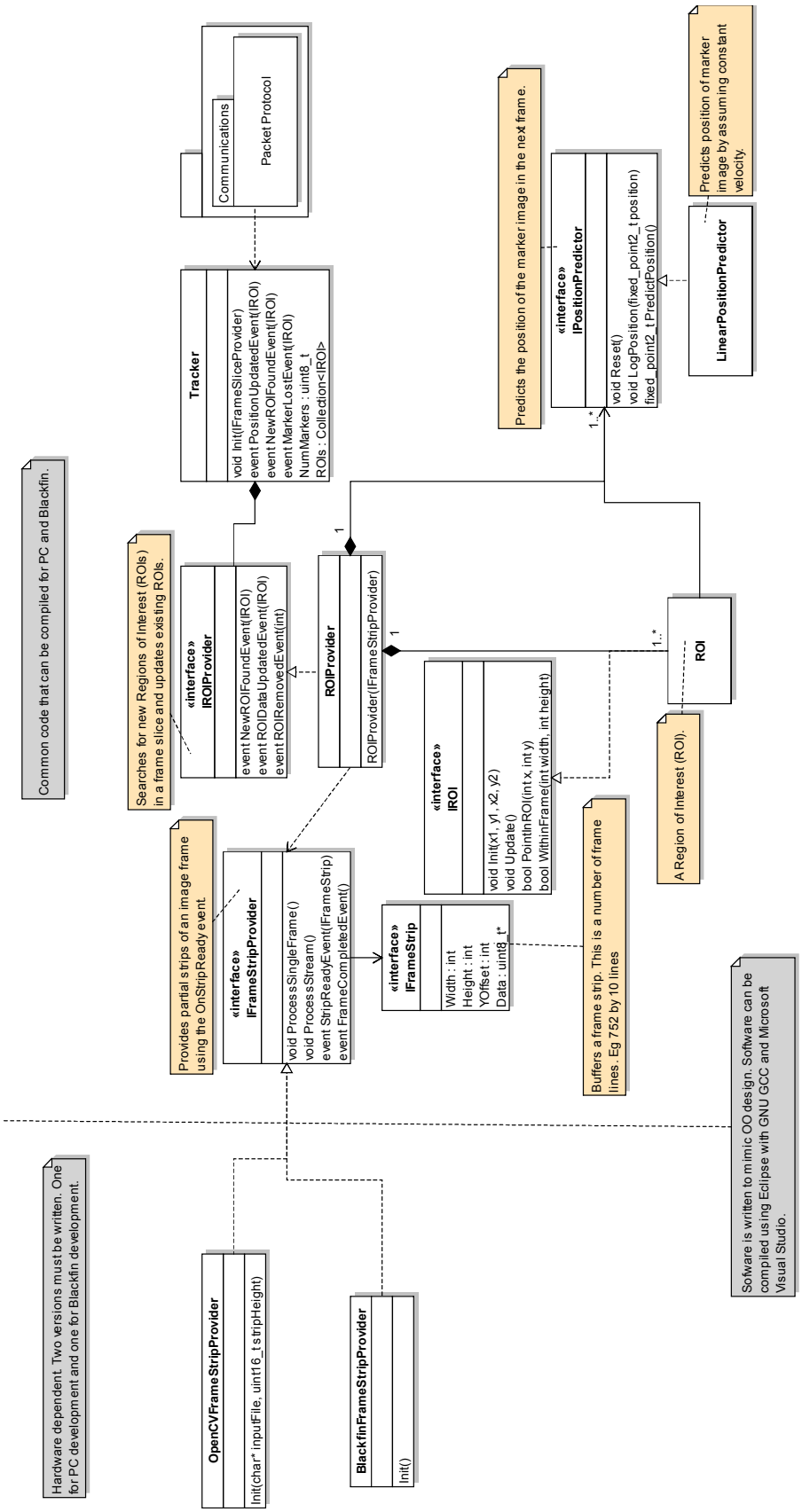


Figure 4.11 : The main functionality of the Black Spot firmware is shown using a UML class diagram.

interface in the figure allows the differing implementation between PC and Blackfin to be ignored. As long as the `BlackfinFrameStripProvider` and `OpenCVFrameStrip-Provider` classes implement the `IFrameStripProvider` interface correctly then the higher level software need not know which class it is using.

The `ROIProvider` class is responsible for discovering new ROIs and tracking them from frame to frame (the segmentation and tracking phases). A `ROIProvider` contains a collection of ROI objects. These are the ROIs that surround the marker images known to the system. A ROI object is associated with an `IPositionPredictor` interface which is used to predict the position of the ROI in the next frame. Due to restrictions in implementation, the objects that implement the `IPositionPredictor` interface are managed by the `ROIProvider` object. The centroid data from the `ROIProvider` are passed onto the `Tracker` object using an event driven mechanism and passed to the communications module and onto the hub. The module and hub communications protocol is described in Chapter 6.

4.4.1 Black Spot Image Sensor Driver

The `BlackfinFrameStripProvider` (see Figure 4.12) implements the `IFrameStrip-Provider` interface and provides frame strips to the firmware when it is running on the embedded system. It is composed of an `IImageSensor` interface and using this interface decouples the frame strip provider from the image sensor implementation. This should make it easier to migrate to other image sensors if desired. The `MT9VX` class implements the `IImageSensor` interface and encapsulates the details of retrieving image data from the sensor. It includes code to manipulate image sensor registers and set different sensor resolutions, shutter periods, and modes.

The Blackfin image sensor driver (the `MT9VX` class) is described below. The driver uses the Direct Memory Access (DMA) features of the Blackfin processor to efficiently copy data provided by the image sensor into the internal SRAM of the DSP. Using the DMA frees the DSP from copying the data and allows it to do meaningful processing in parallel with the DMA. This approach is implemented using double buffering. One buffer is filled by the DMA while the other is used by the firmware. After the first buffer is filled, the firmware uses this one and the DMA writes to the second buffer.

A DMA descriptor chain is initialised with two descriptors. These feed data into the two buffers. The DMA engine begins at the beginning of the chain and works to the end and then repeats. The first descriptor instructs the DMA to write into the first buffer and the second descriptor writes into the second buffer. The buffers are set up so that after a num-

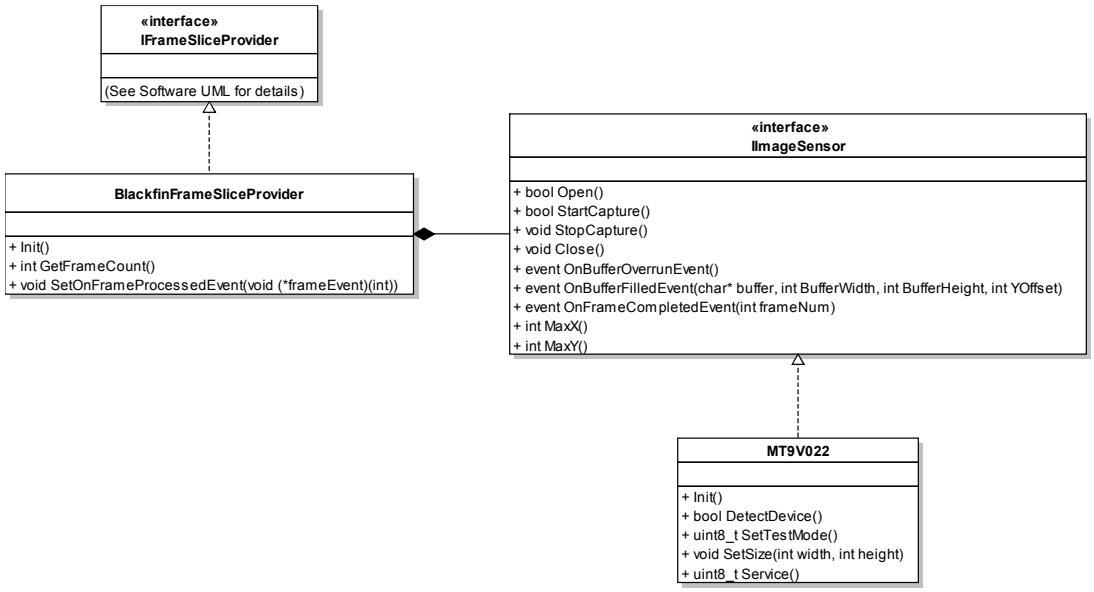


Figure 4.12 Blackfin specific portions of the firmware.

ber of repeats of the chain an entire image has been passed piecewise through the buffers. Figure 4.13 illustrates this scheme.

After the first buffer is initially filled, the DMA immediately begins filling the second buffer. While the second buffer is filled, client code can process the first buffer. This is enabled by subscribing to the events in the `IImageSensor` interface. These events are `BufferFilledEvent`, `BufferOverrunEvent`, and `FrameCompletedEvent`. All processing of the image buffer data occurs in the `BufferFilledEvent`. The consumer of this event must process the buffer data before the next buffer is filled. Failing to do this will result in a buffer overrun and the `BufferOverrunEvent` will be called. In this case the driver ceases processing of image data.

The main loop of the client code calls a service function on each iteration and this enables the `IImageSensor` events to be fired. An alternative approach is to add the service function to a periodically triggered low priority Interrupt Subroutine (ISR) to remove the need for the service routine to be called during the main loop. This could also be achieved by using a Real Time Operating System (RTOS) As the frame is available in strips, all algorithms must be able to operate with only the data available in the current buffer. This complicates the implementation of the algorithms described above.

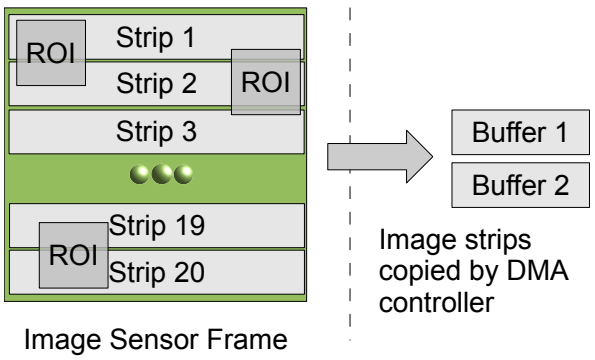


Figure 4.13 The Black Spot double buffering scheme involves copying image frame in parts to two buffers using a Direct Memory Access (DMA) peripheral. One buffer is accessed by the DMA while the other is accessed by the firmware.

4.4.2 Notes on coding conventions

The firmware is written to incorporate OO principles whenever possible. A coding convention was developed to imitate classes in OO languages. Structures are used to represent classes and function pointers used to represent class methods. Object references are pointers to these structures.

The implementation is event driven. Events are implemented using function pointers. While this is not a true event, as only one consumer can subscribe to the event, it is sufficient for this implementation.

In the implementation of the firmware there is not any dynamic memory allocation. This was an implementation decision to improve the robustness of the code. Avoiding using dynamic memory allows the memory requirements of the firmware to be almost entirely determined at compile time. This avoids the situation of algorithms failing due to memory allocations failing.

There is one case where memory is still dynamically allocated and this is for the stack. When a function is called a stack frame is created that holds the previous DSP state before the function call. If too many nested functions are called this could result in a stack overflow. There are not any recursive function calls so this should not be an issue.

4.5 Summary

A basic version of the Black Spot firmware has been designed and implemented with the ability to track the motion of 27 markers during a typical scan scenario at 60 fps. Using a prediction algorithm, the maximum ROI size can be reduced from 70 by 70 pixels to 32 by 32 pixels. The firmware is designed to be compiled using a PC or the Blackfin DSP and the main algorithms can be tested using a PC. Decoupling of the firmware algorithms from the image sensor by way of an interface allows the software to be tested using test videos. Further development should include implementation of the algorithms identified in the future improvements section (Section 4.2.5).

Chapter 5

Black Spot Hardware

The Black Spot is a small camera module containing a processing unit, an image sensor, and a lens assembly. A photograph of this module is shown in Figure 5.1. In the photograph, the lens assembly is the white cylindrical mount and the large black chip is the processing unit (Blackfin DSP). This unit runs the software described in Chapter 4. Modules can be connected together using a shared bus and transmit data to the hub (Figure 5.2). This chapter describes the hardware and the decisions that were made during design and development. Initially a number of different designs were considered and an overview of these are given in Section 5.1. The devices chosen for the design are discussed in Section 5.2 followed by design considerations (Section 5.3), a design overview (Section 5.4), and finally the production of these modules (Section 5.5).

5.1 Design topologies

A number of designs were initially investigated before choosing a design based around a Blackfin DSP and a CMOS image sensor. The base design is shown in Figure 5.3. This shows an image sensor attached to a device marked X. This device provides the clock signal to the image sensor, sends and receives configuration data, and receives image data. Field Programmable Gate Arrays (FPGA), general purpose microprocessors, and DSPs were considered for this device. The advantages and disadvantages of each device are described in the following subsections.

5.1.1 Image sensor and FPGA with RISC soft core

One approach for the camera module would have been to combine a CMOS image sensor and a FPGA with a soft core processor. FPGAs contain many programmable logic elements that can implement AND, OR, and NOT gates [73]. These logic elements can be combined

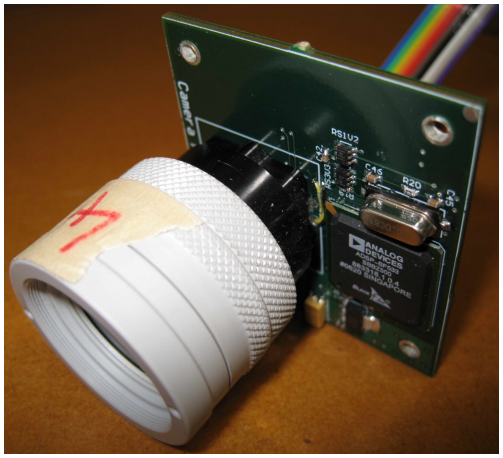


Figure 5.1 A Black Spot module is shown. The module includes a lens assembly (left of figure), image sensor, and Digital Signal Processor (DSP). The module is capable of tracking the locations of markers within its FOV.

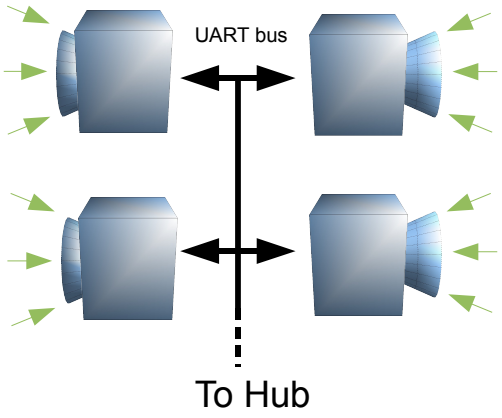


Figure 5.2 Multiple camera modules communicate with the hub using a multidrop UART bus.

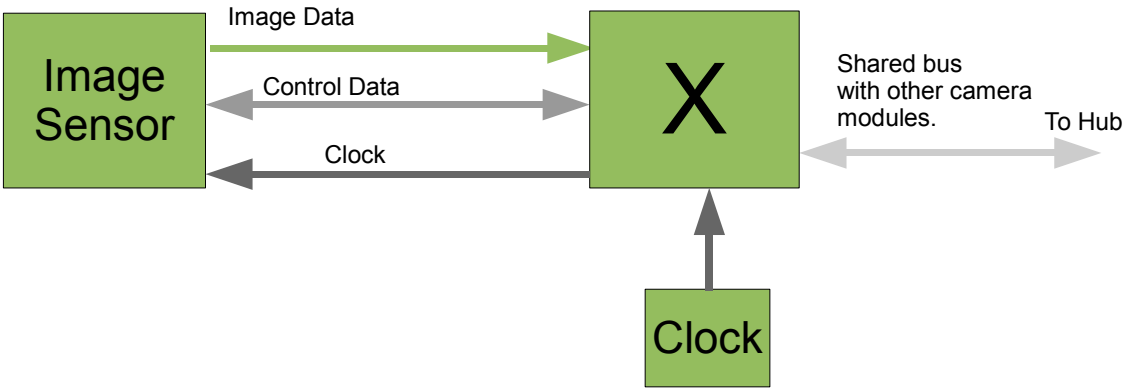


Figure 5.3 Designs considered use a image sensor attached to either a FPGA with a RISC soft core, a general purpose microprocessor, or a DSP.

to produce complex high level modules and could be used to build very fast algorithmic blocks for the camera module. A soft core processor is a processor that is embedded into the logic elements of the FPGA. Standard C code can be executed on the soft core and custom blocks can be executed from the C code [74]. This allows very specific optimisations to be made by implementing some algorithms using the FPGA's programmable hardware and developing other parts in software. This approach is powerful as it fuses the speed of a hardware based approach with the flexibility of a software approach. Although this method has great potential it was discarded as it was felt that the learning curve for using an FPGA could be greater than other methods considering the author's experience with these devices.

5.1.2 Image sensor with low power general purpose microprocessor

Another option was to connect a low power general purpose processor (such as a microcontroller based around an ARM core) to an image sensor, however, this approach was discarded as it was thought that many low power processors do not have the processing power required. Also, many microprocessers do not have suitable peripherals to allow connection to image sensors.

5.1.3 Image sensor with DSP

A final option was to use a DSP and connect this directly to an image sensor. This allows the performance of DSPs to be leveraged. Careful choice of DSP is required, however, to find a DSP that can be connected directly to an image sensor. A Blackfin DSP from Analog Devices Inc. was chosen. The choice of model is described in Section 5.2.1.

5.2 Device selection

The option chosen was to use a DSP connected to a CMOS image sensor using either external SDRAM or only internal SRAM. DSPs are well suited to processing large quantities of data quickly and often have special instructions to make processing more efficient that standard microprocessors do not have. Using a DSP is a good compromise over using custom hardware blocks programmed into a FPGA.

5.2.1 DSP selection

The Blackfin family of DSPs from Analog Devices Inc. were considered. Blackfin DSPs sit somewhere between traditional DSPs and general purpose standard microprocessors. They have many of the instructions found in traditional DSPs such as a one clock cycle multiply and accumulate operation but, unlike more traditional DSPs, contain an abundance

Name	Max. Clock (MHz)	Max. Program SRAM (bytes)	Max. Data SRAM (bytes)	Notes
Blackfin BF531	400	32K	16K	
Blackfin BF532	400	48K	32K	
Blackfin BF533	600	80K	64K	
Blackfin BF534	500	64K	64K	
Blackfin BF536	400	64K	32K	Ethernet MAC
Blackfin BF537	600	64K	64K	Ethernet MAC
Blackfin BF561	600	32K / core	64K /core	Dual core, 128K bytes L2 SRAM

Table 5.1 A selection of the Blackfin family that Analog Devices lists as supported by GNU tools is shown here. The Blackfin BF533 has the greatest amount of internal L1 SRAM of this selection.

of I/O pins and peripherals. For example, the Blackfin BF533 has up to 16 general purpose I/O lines and contains a UART module, Real Time Clock, and other specialised peripherals. Many Blackfin chips are well supported by the GNU GCC open source compiler. They are designed for video applications and have custom instructions for processing video data as well as a bus interface suited to connecting to CMOS image sensors. These processors have a strong support community on the internet. In addition to this an application note from Analog Devices describes connecting a CMOS sensor to the Blackfin [75] and one of the author's supervisors, Dr Michael Hayes, had prior experience using these chips.

There are many processors in the Blackfin range and a number are supported by the GNU GCC compiler at the time of part selection. Table 5.1 lists these chips and their maximum clock rates, and memory capacities. Since part selection, Analog Devices has introduced many more processors to this family and the GCC compiler is increasing support for them. Of particular note is the Blackfin BF533 and the Blackfin BF561. At the time of writing, the Blackfin BF533 has the most internal high-speed "Level 1" (L1) SRAM of the family[1] and this is attractive as it could lead to a system that does not require SDRAM. The Blackfin BF561 is the only chip in the family that has two processing cores in the same physical package. Using this chip could allow a greater range of algorithms to be implemented due to the extra processing facilities, however, designing algorithms to run in parallel on separate processors is not trivial and would add considerable complexity to the software.

For the prototype, the Blackfin BF533 was chosen primarily due to the amount of RAM that it has.

[1]The BF561 has more SRAM but a portion of this is classed as "Level 2" SRAM and runs at half the clock speed of L1 SRAM and with longer latencies.

5.2.2 Image sensor selection

Whilst deciding to use a Blackfin DSP, a suitable image sensor was also considered. CCD image sensors and CMOS sensors were considered. CCD sensors traditionally have better noise performance compared with CMOS sensors [76] but require an extra stage to convert their analog image data to digital data so that it can be used by the board's processor. Due to this CMOS sensors were favoured over CCDs.

Aptina Imaging (formerly Micron) produce a range of sensors that were of interest. The MT9V022 sensor [77] is one of these and is a 752 x 480 Wide VGA sensor. As noted, Analog Devices publish an application note [75] that discusses interfacing this sensor to their Blackfin chips and this was an important factor in the decision to use this sensor. Also of importance is the relatively high frame rate of 60 fps. At the time of selection many chips provided 15 or 30 fps output but fewer output at 60 fps or higher.

A greater frame rate reduces the processing requirements for the tracking algorithms. This is because the marker images cannot move as far between frames and so smaller ROIs can be used to produce the same tracking performance. For example, an increase in frame rate of $2\times$ means that a marker can only move half as far per frame. This means a ROI of half the width of the original ROI can be used. Although this results in twice as many ROIs to process over time, the number of pixels to analyse is lower. This is because the number of pixels in a ROI is equal to the square of the width of the ROI and the width of a ROI required to give equivalent performance decreases proportionally with the frame rate. This results in fewer pixels being analysed and this is an important factor.

Another important feature of the sensor is that it implements a global shutter. Essentially, a global shutter means that all of the data in an image frame is available to the processor at a given time. In a rolling shutter system, data from the current frame can be mixed with data from a previous frame leading to a tearing or wobbling effect during high motion. These artefacts could corrupt data from the tracking system and, therefore, it is important that a global shutter is used. At the time of selection many chips implemented a rolling shutter.

The MT9V022 sensor is available without a Bayer colour filter thereby allowing gray scale images to be captured with complete intensity data for each pixel. This is important in this application as a colour filter could warp the data presented to the centroid algorithm and colour data is not required. Many other models were not available in monochrome.

A later model of the image sensor, the MT9V023 [78], is also available. Both sensors are similar but initially Aptina marketing material indicated that the later model would require approximately half as much power as the older model (this claim was later revoked). The newer sensor is pin compatible with the older sensor and the major difference is the

register set. The MT9V023 registers are a super-set of the older sensor's but are similar. The MT9V023 has two banks of registers and the sensor can be switched rapidly between each bank. Both sensors were used during production of the board.

Data from these sensors are output using a 10 bit parallel bus and also using a serial Low Voltage Differential Signaling (LVDS) bus. The chips provide a Two Wire Interface (TWI) bus for control of parameters such as software reset, shutter period, automatic gain control, automatic exposure control, and frame size. The sensor can be clocked at a maximum of 27 MHz and as low as 13 MHz to produce images at approximately 30 fps. The data sheet for the MT9V023 claims sensitivity to light of at least 55 dB and at least 100 dB using a compander mode. The chips are designed for a 1/3" optical format allowing for smaller optics than some other chips considered that use a 1/2" format. At the time of selection, the sensors were available in a BGA package, however, a similar model (MT9V032 [79]) is now available in a slightly more prototype friendly CLCC package. They are most responsive to light with a wavelength of approximately 600 nm with slightly lower efficiency at the LED wavelength of 510 nm.

At the time of selection both chips were available from the American electronics supplier Digikey for approximately $30 US with a minimum order quantity of 1 unit. However, at the time of writing Digikey does not stock these parts giving weight to moving to the CLCC packaged part that remains available.

For the prototype, the MT9V023 sensor was used on two of the three prototypes produced. The MT9V022 sensor was placed on the other prototype board.

5.3 Design considerations

During the design of the camera module, parameters such as size, power requirements, cost, and interference issues were considered. The modules will be attached to the head of a laser scanner, therefore, they must be small and light so that the scanner does not become unwieldy to use. Since this design is a prototype the requirement of producing a small design can be relaxed, however, it is important that the design is compact enough such that the prototype can be developed and reduced in size in the future.

Each module must consume little power to minimise the heat produced. Since there will be many modules in the system the overall power consumption will be many times greater than the power consumption of a single module. Another design goal is to keep the part count low primarily to reduce the production cost and make routing the PCB easier. The ease with which the PCB can be routed is also affected by the high speed signals in the design, i.e., the fewer high speed buses there are, the easier it is to route.

5.3.1 Estimate of required memory

Another quantity that was considered in the design of the board was the memory required to run the firmware algorithms. In a Modified Harvard Architecture microprocessor such as the Blackfin, system memory is split into program memory and data memory. Program memory contains the instructions that are executed by the processor while data memory contains the non-executable data such as program variables.

Data memory estimate

The image processing algorithms in the firmware operate on portions of the image from the image sensor using a double buffering scheme. To do this the image data must be buffered in RAM. To buffer an entire image can require a lot of RAM for an embedded system. For example, an 8 bit gray scale 752 x 480 pixel image requires 360960 bytes. While this is tiny compared with the RAM available on personal computers, it is sizable for an embedded system and would require external SDRAM and potentially a larger PCB. For a system using only the internal SRAM of the Blackfin an entire image frame cannot be stored at one time and each image must be processed in parts. For example, a frame could be broken into 24 parts giving a buffer size of $752 \times 480/24 = 18$ K bytes, or it could be broken into 32 parts giving a buffer size of 11 K bytes.

There is a greater overhead in terms of computing power required when using small buffers as they must be switched more frequently (double buffering), however, smaller buffers lead to a lower processing latency. This is because the system must wait for an entire buffer to be filled before it can process it even though a marker image may only cover a small portion of the buffer. The problem becomes a trade-off between the memory required for the buffers versus the overhead of switching buffers. The assumption during design of the software is that in a system using only SRAM all available memory (64 K bytes using the Blackfin BF533) would be used and the size of the buffers would be adjusted so that the application fits into RAM. A buffer size of $752 \times 24 = 18$ K bytes has proven to be practical. As the system uses double buffering, two buffers of 18 K bytes are created. All processing occurs in place and therefore there is not any need to copy the ROI data to separate buffers. This brings the total image buffer size to 36 K bytes.

Program memory

Estimating program memory is more difficult than estimating data memory. It is believed that the 80 K bytes of internal SRAM available in the Blackfin BF533 is enough for the application, however, using external SDRAM could make the firmware design easier as the constraint to keep the program code small would be removed. With external SDRAM the program size could increase significantly towards the maximum size supported by the

Blackfin BF533 of 132 M bytes.

5.3.2 Estimate of required processing capabilities

It was difficult to accurately predict the computational requirements of the camera module, however, a rough estimate of the bare minimum processing power was useful during design. The most intensive part of the firmware is calculating the centroid of a ROI. A 32 by 32 pixel ROI contains 1024 pixels that need to be processed. In Chapter 4 it is estimated that this requires approximately 41.5 Mips (Equation 4.2). The Blackfin BF533 can operate at up to 600 M Hz giving a total of 600 Mips. The device also allows execution of two 16 bit instructions in parallel with one 32 bit instruction under some circumstances. Optimisation of the centroid algorithm to use efficient instructions on the Blackfin could also increase the processor's performance. Clearly, as far as speed is concerned, the Blackfin is a good choice for this application.

5.3.3 Estimate of power requirements

An estimate of the power required for the main components of the module is given in Table 5.2. This is based on data found in the datasheets for the Blackfin BF53x family [80], image sensor [78], and linear regulator [81] and assumes the Blackfin is in its active state all of the time. The assumption that the Blackfin is active 100% of the time leads to a power estimate that represents a worst case scenario. As shown, the Blackfin and image sensor have similar power requirements. Using the current drawn by both these components the power dissipated by the 3.3 V linear voltage regulator on the board was calculated. This was calculated using the voltage drop between the 5 V supply and the 3.3 V regulator output given by

$$P_{REG} = I_{REG}(V_{IN} - V_{OUT}), \tag{5.1}$$

and the current flowing through the regulator, i.e.,

$$I_{REG} = I_{DSP} + I_{SENSOR}. \tag{5.2}$$

This gives

$$I_{REG} = 240 + 97 = 337\,\mathrm{mA} \tag{5.3}$$

which leads to

$$P_{REG} = 0.337(5 - 3.3) \approx 573\,\mathrm{mW}. \tag{5.4}$$

This is a large percentage of the power requirements and indicates using a more efficient switching regulator would be a good option.

Part	Power Estimate (mW)
Blackfin BF533	288 (Vddint = 1.2V, f_{CCLK}= 500MHz from Blackfin BF53X family data sheet.)
MT9V023	320
3.3V Linear Regulator	573 (Power calculated using current drawn by Blackfin and CMOS sensor.)

Table 5.2 An estimate of the power required by the major components of the system is given in this table and equates to 1181mW.

5.3.4 Thermal considerations

Using the estimated power required in the previous section, the heat produced by the three main components can be calculated. This calculation will illustrate whether the specified operating temperature of the module given in Chapter 2, i.e. an ambient temperature of between 5 and 35°C will result in the three components exceeding their maximum operating temperatures as given in their datasheets. To determine this the following assumptions are made. The module is assumed to be in a steady state with regard to temperature. This assumption is based on a scenario of constant power input and constant ambient temperature. The calculations assume passive cooling. Only the Blackfin DSP, image sensor and linear voltage regulator are considered. It is assumed all other components will dissipate negligible amounts of power and can be ignored.

First the Blackfin BF533 is considered. The Blackfin BF53x family datasheet [80] gives a maximum junction temperature of 125°C . A formula for calculating a first order approximation of the junction temperature based on ambient temperature (T_A), power dissipation (P_D), a thermal coefficient that varies depending on the chip package type, and air flow (θ_{JA}) is given in the data sheet. This is

$$T_J = T_A + (\theta_{JA} \times P_D). \tag{5.5}$$

The chip package is a 169 BGA package and the assumption is zero air flow leading to a thermal coefficient from the data sheet of $\theta_{JA} = 22.8°\mathrm{C/W}$. For the maximum ambient temperature of 35°C (given in the specification in Chapter 2) the junction temperature from the formula is

$$T_{\mathrm{JDSP}} = 35 + 22.8 \times 228 \times 10^{-3} \approx 40°\mathrm{C}. \tag{5.6}$$

This is well below 125°C .

The Aptina MT9V023 data sheet [78] gives an operating temperature of at most 85°C. A maximum junction temperature is not given in the data sheet and there are not any thermal coefficients given so it can only be assumed that the figure given represents the maximum

ambient temperature. Once again 35°C is much lower than 85°C.

The third chip to analyse is the linear regulator. The regulator chosen is a ADP3338AKC-3.3 [81] from Analog Devices, Inc. and its datasheet gives a maximum junction temperature of 150°C and a thermal coefficient of $\theta_{JA} = 62.3°\mathrm{C/W}$. Using Equation 5.5 this gives

$$T_{\mathrm{JREG}} = 35 + 62.3 \times 573 \times 10^{-3} \approx 71°\mathrm{C}. \tag{5.7}$$

This is also well below the maximum quoted.

This brief analysis shows that the camera modules can operate at an ambient temperature of 35°C with passive cooling without exceeding the maximum operating temperatures given in the three main components data sheets.

5.3.5 System with external SDRAM

With the processor family and image sensor chosen a decision was made whether to use an SDRAM external to the Blackfin (Figure 5.4) or to rely solely on the internal SRAM available. Using external SDRAM has advantages as it allows buffering of a full image frame in memory that is not possible with limited internal SRAM. This is particularly useful for debugging as it allows an entire image to be captured and relayed to software for analysis. It also simplifies image processing algorithms as they can operate on data from anywhere within the entire image. Without SDRAM only portions of an image can be processed at a given time requiring algorithms to be adapted to work in this manner. These two reasons make a strong case for using SDRAM.

However, there are disadvantages to using external SDRAM. These are the added hardware complexity, increased potential for corruption of data due to noise, increased part count, potentially increased size, and increased cost. For this prototype the part count, size, and cost are of lesser importance. Complexity and noise are important issues, however.

By adding SDRAM complexity is increased. Firstly, routing the PCB is more difficult as a SDRAM requires many tracks for transferring data between the Blackfin. The placement of these extra lines on the PCB makes routing the PCB more difficult and time consuming. Also, the SDRAM requires a 133 MHz clock. A high speed clock such as this is a source of switching noise and the PCB would require careful design to avoid this noise coupling with data lines on the board. This again makes routing more difficult and time consuming and increases the chances of producing a board that cannot be used. Finally, SDRAM running at 133 MHz is slower than the internal SRAM meaning that the processor may have to wait to process data and this could slow the system down. Careful design and use of the caching

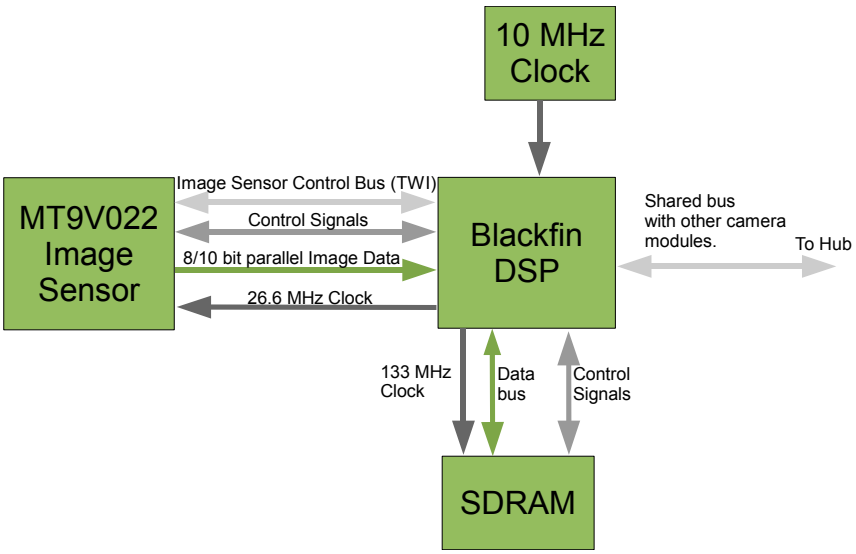


Figure 5.4 A possible design consisting of an image sensor, Blackfin DSP, and external SDRAM.

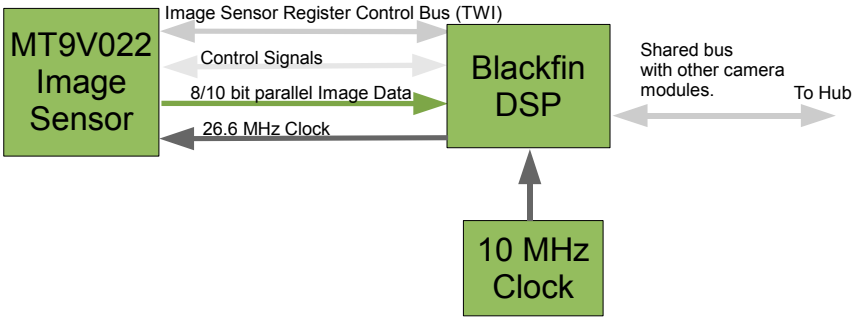


Figure 5.5 The approach chosen uses a CMOS sensor and Blackfin DSP with internal SRAM.

facilities of the Blackfin could mitigate this at the expense of designing the software to work with partial image frames. Due to the limited time available to produce a prototype, the author's hardware design experience, and the need to produce a board with the greatest potential to work correctly, external SDRAM was discarded.

5.3.6 System with internal SRAM

A system using the Blackfin BF533 DSP with internal SRAM only (Figure 5.5) was chosen for the reasons discussed above.

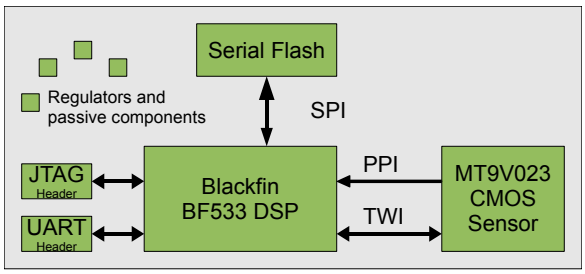


Figure 5.6 A block diagram of the main camera module components.

5.4 Design overview

A schematic was designed that allows the Blackfin to be interfaced to the image sensor and data from the Blackfin to be sent to the hub. This involved choosing parts to supply the Blackfin and image sensor with the power supplies, clocks, and data buses that they require. It also requires a method to allow the boards to be programmed, debugged, and connected to the hub. The board design was based around the open source BlackfinONE project [82]. A block diagram of the board's main components is shown in Figure 5.6. This diagram shows the Blackfin connected to a serial flash chip, image sensor, headers, and surrounded by supervisor and passive components. The flash chip stores the program code that the Blackfin executes. This is loaded into the Blackfin by an on-chip bootloader during start-up. On the left of the figure is the JTAG debugging header. This allows code to be loaded into the Blackfin's internal SRAM by a PC bypassing the flash during debugging of the firmware. Below the JTAG header the communications header is shown. The hub is interfaced to each camera module through the communications header. This connector also carries power and timing signals. The interfacing of the parts with the Blackfin are discussed in more detail in the following sections.

5.4.1 Image sensor interface

The image sensor is connected to the Blackfin via a parallel interface and also via a two-wire interface (TWI). The parallel interface is used to transfer the image data from the sensor to the processor. This comprises a 10 bit data bus and five timing signals. Only the 8 most significant bits are connected to the Blackfin to reduce the memory required. The DMA controller in the Blackfin is capable of transferring 16 bit data and, therefore, could transfer 10 bit image data. However, this would require twice as much memory as 8 bit data because each 10 bit word would be stored using 16 bits within the Blackfin's memory. The TWI is used to transfer sensor initialisation, control, and status information. This bus is used by the processor to set the image sensor resolution, shutter period, and other sensor parameters.

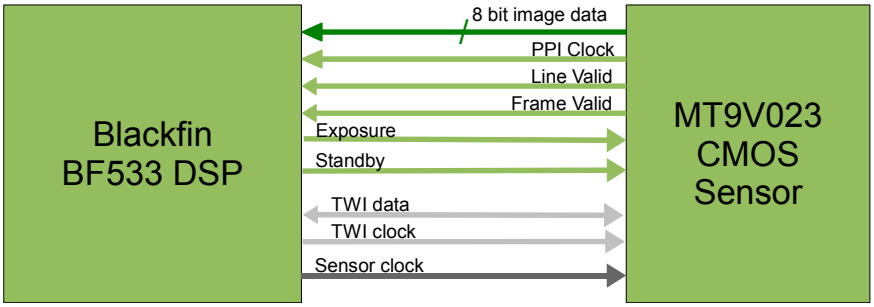


Figure 5.7 The Blackfin DSP is connected to the image sensor by an image data bus and control signals and a two wire data bus for control of the sensor by the DSP.

Figure 5.7 shows the Parallel Peripheral Interface (PPI) bus and TWI bus connecting the image sensor and Blackfin in more detail. The parallel buses five timing signals are the sensor clock, PPI clock, line valid, frame valid, and exposure signals. The sensor clock is the main clock for the sensor. This signal dictates the maximum frame rate of the sensor. It is 26.6 MHz for the maximum 60 fps that the sensor supports. Lower frequencies lead to lower frame rates. This signal is controlled by a line output from the Blackfin and the frame rate can be set by the DSP.

The timing signals are used to synchronise the image frames with the Blackfin. The PPI clock is the return signal from the image sensor and is derived from the sensor clock signal. This is used to clock the parallel peripheral interface on the Blackfin. The line valid and frame valid signals instruct the PPI port as to when the bus contains valid image data. This is used internally by the Blackfin's DMA controller to copy image data to memory.

The TWI bus is a multi-drop bus compatible with the Inter-Integrated Circuit (IIC) bus from NXP Semiconductors [83] (formerly Philips Semiconductors). The bus is half-duplex and the data lines and clock are shared between upstream and downstream devices. The bus is based around a master and slave configuration. In this application, the Blackfin is the bus master and the image sensor is the slave. The slave drives the bus only when the master has requested to read data from the slave or when the master is writing to the slave and the slave is acknowledging the written data. The BF533 does not have a hardware TWI so the TWI lines are connected to general purpose I/O lines and the interface is implemented in software.

5.4.2 Flash chip interface

The Blackfin processor is connected to a Serial Peripheral Interface (SPI) flash memory chip. The serial flash chip is used to store program code so that the camera module can run without a PC attached. The chip is programmed using software running on a PC and a hardware JTAG debugger interface. In theory it is also possible to program the flash via

the Soft Hub if this functionality were programmed into the Soft Hub.

5.4.3 Programming and debugging the Blackfin

The Blackfin has an on-board JTAG interface that is exposed using the JTAG header on the PCB. The 7 by 2 pin JTAG header allows program execution to be controlled and the program variables to be studied. This is achieved using the software development environment and a hardware adapter. The firmware can be loaded into the camera module for execution by either using the hardware JTAG adapter or from the flash memory. During development the firmware is loaded using the JTAG adapter as this allows software revisions to be loaded and debugged quickly. Once a stable version of firmware is ready it is loaded into the flash memory on the PCB. The flash memory is a non-volatile storage meaning that the memory contents remain intact without continuous power. Powering up the module results in the program code being copied from the flash to the Blackfin's internal program memory by a bootloader. This is a small program that resides in a boot ROM in the Blackfin designed specifically for initialising the Blackfin. Execution of the code begins once this is complete. The JTAG adapter chosen is the ICEbear from the Swiss company 'Section5' [72]. This debugger integrates with the software environment described in Chapter 4 and connects via USB to the JTAG header.

5.4.4 Power supplies and clocks

There are three power supplies and two major clock signals present in the design. The power supplies are 5 V, 3.3 V, and 1.2 V. The design is powered using 5 V and this is provided through the communications header. This is regulated to 3.3 V by a linear regulator. A buck converter controlled by the Blackfin regulates this again to 1.2 V. The Blackfin requires both 3.3 V and 1.2 V to operate. The I/O pins of the Blackfin operate at 3.3 V logic levels while internally the core operates at 1.2 V.

The Blackfin is clocked using a 10 MHz crystal. Internally this is scaled to provide a core clock frequency of

$$C_{CLK} = 10 \times 10^6 \times 43 = 430\,\text{MHz}. \tag{5.8}$$

This ensures that the Blackfin can provide a clock signal to the image sensor. The Blackfin peripherals operate at a lower frequency, referred to in the data sheet as the system clock frequency. The core clock is divided by 4 to give the system clock of

$$S_{CLK} = \frac{C_{CLK}}{4} = 107.5\,\text{MHz}. \tag{5.9}$$

The system clock is used to clock a Blackfin timer peripheral that provides a further division by 4. The output of the timer is used to clock the image sensor and is a square wave

with a frequency

$$I_{CLK} = \frac{S_{CLK}}{4} = 26.875\,\text{MHz}. \tag{5.10}$$

The multipliers and divisors are programmable and can be controlled in the firmware. The values chosen ensure that the image sensor is clocked as close to its maximum clock rate as is possible.

5.4.5 Communications with the hub

A number of camera modules can be connected to the central hub by a shared communications bus. The bus transceivers are standard Universal Asynchronous Receiver/Transmitters (UARTs) with 3.3 V CMOS logic levels. The communications protocol and timing are described in Chapter 6 and only the hardware aspects are noted here. The bus is shared in such a way that data transmitted by the hub is received by all of the camera modules. Data sent by any camera module is only received by the hub but the transmission line is shared by all camera modules. It is possible for any camera module to drive the bus at any time and this means a mechanism is required to avoid collisions on the bus when data is sent to the hub. Unfortunately the Blackfin UART transmit line cannot enter a high impedance mode and therefore drives the bus continuously. This problem is overcome by the bus sharing circuit.

The hardware side of this mechanism protects the camera modules and stops excessively high currents flowing. It is a simple arrangement using a Shottky diode and resistor (Figure 5.8) and enables camera modules to enter a high impedance mode when not driving the bus. The resistor is used to pull the bus to 3.3 V and the Shottky diodes are in series with the Blackfins' transmit lines. When a Black Spot module sets a logic zero the diode is forward biased and current flows pulling the bus low. In this arrangement if any module drives a logic zero then the bus registers a zero. When a module sets a logic one the diode is no longer forward biased and the bus sits at 3.3 V due to the pull-up resistor.

The communications header is shown in Figure 5.9. It is a standard 0.1 inch pitch 5 by 2 pin header. The header provides the Black Spot module with power (pin 9), communications with the hub (pins 5, 7), bus in use signal (pin 3), and flow control (pin 1). The remaining pins are grounded. Pin 2 is used as a key and is removed from the header. The corresponding plug has the hole that corresponds to pin 2 filled and this assures that the plug is inserted in the correct orientation.

5.4.6 PCB routing

The printed circuit board was routed using four layers. This was done for two reasons. Firstly, the greater the number of layers that a board has, the easier it is to route. Secondly

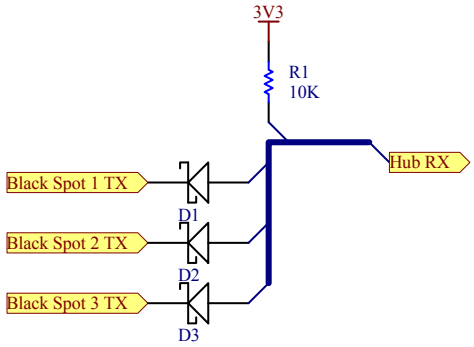


Figure 5.8 The Blackfin transmit line is protected against two or more modules using the bus at once by a Shottky diode and a resistor.

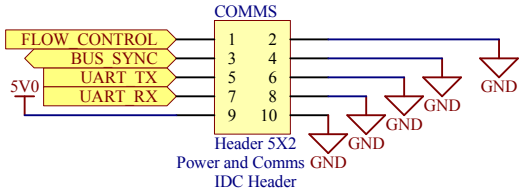


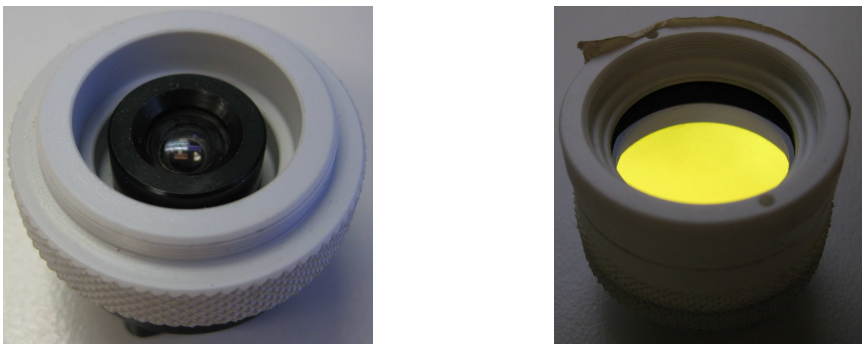
Figure 5.9 The communications header provides the Black Spot module with power and communications with the hub.

a four layer board means that the two inner layers can be used for power and ground planes. This helps to reduce switching noise from high speed signals by providing inter-plane decoupling capacitance. The four layers from top to bottom are a top routing layer, an internal power plane, an internal ground plane and a bottom routing layer. The power plane is broken into 5 V, 3.3 V, and 1.2 V areas. The Blackfin I/O operates at 3.3 V so the majority of the power plane is 3.3 V. Care was taken to ensure that the high speed signals travel as short a distance as possible on the PCB. The image data bus was also routed with each bit next to each other as these signals are likely to change state together and, therefore, probably will not affect each other. The 3.3 V I/O signals on the top and bottom layers were routed so that they run over the 3.3 V plane area, again to reduce noise.

5.4.7 Optical design

The optical components of the system consist of a miniature S mount lens, a custom CS mount lens adapter, and an optical filter. The miniature lens is a small lens supplied by Edmund Optics and has a focal length of 6 mm. It screws into the CS mount lens adapter (Figure 5.10a). A narrow-band green filter from Edmund Optics is attached to the adapter (Figure 5.10b). This filter has a bandwidth of 10 nm (P/N 510FS10-25). The advantage of using a filter is that the SNR can be improved by attenuating light at wavelengths other than LED's predominant wavelength. A disadvantage with using the filter is that the brightness of objects at the edges of the FOV is severely reduced.

A 6 mm miniature lens has been chosen for the design. The lens must be able to focus on an LED in the range of 0.5 m to 3 m. As the distance to the LED is much larger than the focal length, the LED is approximated to be at infinity and the thin lens equation (Equation 3.3 presented in the Chapter 3) states that an image is formed at the focal length of the lens, i.e., 6 mm.



(a) (b)

Figure 5.10 (a) The lens screws into the CS mount adapter. (b) The filter attaches to the CS mount adapter.

Focusing the lens

To focus the lens it is rotated inside the CS mount while images are captured using the camera module firmware. When out of focus, the LEDs appear much larger than they are. Rotating the mount towards focus reduces the LED image size. The minimum LED image size is when the LED is in focus. The lens for each module must be focused before the module can be used.

5.5 Production

The PCB is a 50 mm $\times$ 55 mm board and contains approximately 80 parts. These parts include the processor, image sensor, regulators, decoupling capacitors, and resistors. The top side of the board contains the image sensor, Blackfin chip, 3.3 V regulator, supervisory components, and 10 MHz crystal (Figure 5.11a). The bottom of the board (Figure 5.11b) contains the flash chip, Blackfin core voltage regulator, and the communications and debugging headers. There are also a number of test points for debugging using an oscilloscope.

The PCB was manufactured by Advanced Circuits (4pcb.com) in the United States of America. To reduce the cost of manufacturing and part placement the PCB was produced on a panel as an array of 3 by 3 individual boards (Figure 5.12). The outlines of the boards were routed so that the boards can be snapped apart after component placement. A solder paste stencil was also produced by the same company. Most components were sourced from the American electronic supplier Digikey (www.digikey.com).

The processor and image sensor come in Ball Grid Array (BGA) packages. These cannot reliably be placed and soldered by hand. A PCB assembly company (Assembly Specialists

Ltd.) was used and they placed and soldered these parts. To do this, firstly, the solder paste stencil was used to place the solder onto the BGA pads on the PCB. The BGA parts were then machine placed. To allow this to be done, fiducial markers were added to the PCB in the design phase. This enables the machine vision system to automatically place the parts in their correct locations. The panel containing the three loaded boards was placed in a vapor phase oven that heated the solder until the BGA parts were fixed to the board.

The remaining parts were placed by hand at ASL. The resistors and capacitors were chosen specifically to be large enough to solder by hand and are 1.6 mm x 0.8 mm (0603) parts. As the boards were tested it became apparent that modifications were required. These usually involved cutting tracks and rewiring parts. Modifications are listed in Section 5.5.1.

5.5.1 PCB modifications

Throughout the development process modifications were made to the original schematic to fix errors in the design. There schematic changes lead to PCB modifications. An overview of these modifications are described here with detailed diagrams in Appendix D.

Buck converter modification

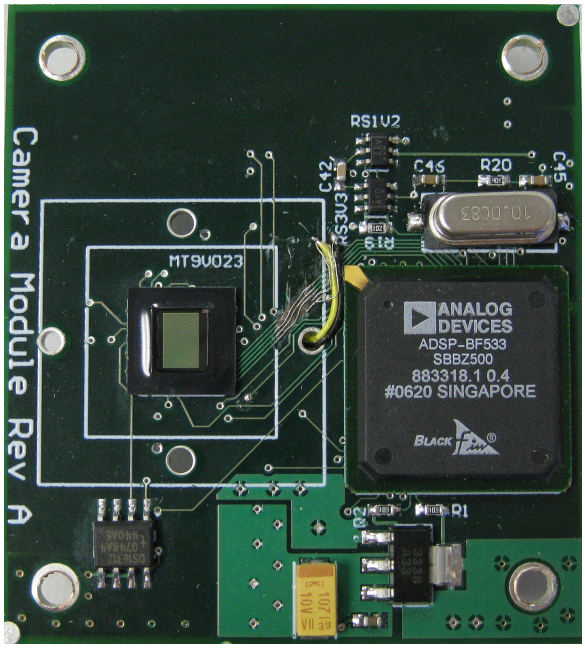
The original schematic contained an error that caused a short between the 5 V power rail and ground. This was due to an error in the buck converter circuit. The board did not function at all before this modification.

Flash select line

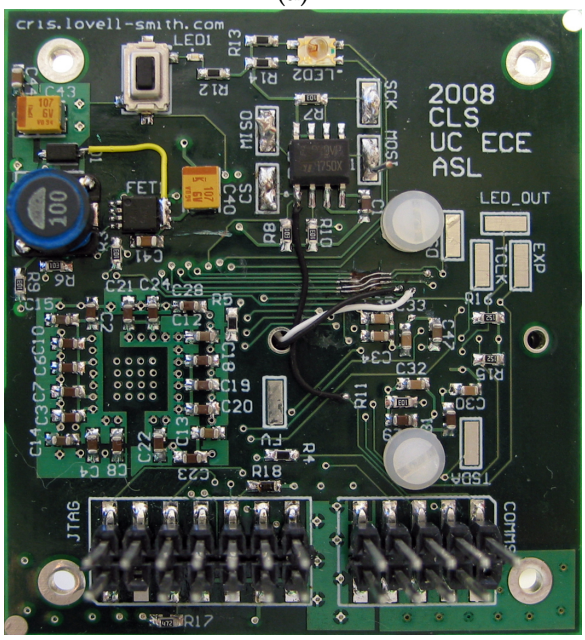
The Blackfin requires the flash select line to be connected to processor flag PF2. Initially this was connected to a different processor flag. This resulted in the board not being able to load program code from the flash.

Image data bus modification

Initially the eight least significant bits of image data were connected to the lower eight bits of the PPI port and the two most significant bits were connected to bits 8 and 9 of the PPI port. While this is not a problem if 16 bit reads are used on the Blackfin it does become a problem if 8 bit reads are used. During development it was decided that 8 bits of image data were sufficient for processing. A better scheme then is to route the 8 most significant bits into the lower bits of the PPI port allowing them to be read in 8 bit read mode. A modification was made to do this.



(a)



(b)

Figure 5.11 (a)The top side of the Black Spot PCB. (b) The bottom side of the Black Spot PCB.

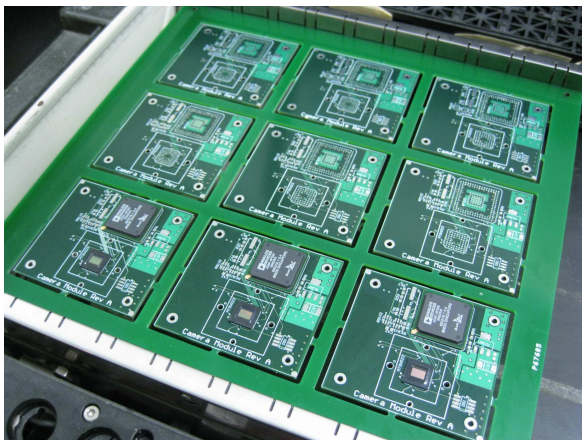


Figure 5.12 The PCB panel contains 9 individual Black Spot boards. Here the front three have been loaded with the BGA parts.

Image sensor reset

A minor error in the RC circuit meant that the reset line was not being cleared as per the specification at start-up. This did not appear to affect the functioning of the system.

UART transmit modification

The Blackfin BF533 cannot tristate the UART transmit line and this lead to problems sharing the bus between camera modules. As described, a reverse biased Shottky diode was used to replace the current protection resistor initially designed on the transmit line. This was coupled with a pull up resistor on the bus shared between all camera modules.

5.6 Summary

The first revision of the camera module hardware design has produced a small module of dimensions 50×55 mm with CS Mount Lens holder and the ability to add lens filters. The board uses the Analog Devices Blackfin BF533 processor and the Micron MT9V022/23 image sensor. It can be programmed using the ICE Bear JTAG debugger and the GNU tools that come with the debugger. Program code can be stored in the non-volatile flash memory and is automatically loaded into the Blackfin for execution when the chip is powered on. Further design details are given in the appendices. The Black Spot schematic is given in Appendix A. Appendix D details the modifications required in order for the PCB to match schematic. The unmodified PCB is shown in Appendix B and the PCB placed in a panel array for production is shown in Appendix C. The Black Spot Bill of Materials (BOM) is given in Appendix E.

Chapter 6

Communications

This chapter describes communications between the Black Spot modules and the Soft Hub. This is a key element in the system as it delivers the data that the pose estimation algorithm requires to the Soft Hub. The need for timely delivery of this data places requirements on the communications system and these are described below.

The system comprises two layers and these are a physical layer and a transport layer. The physical layer consists of the physical connections between the camera modules, i.e., the physical bus and the bus transceivers. The physical layer must allow multiple camera modules to be connected concurrently; it must have sufficient bandwidth to allow the necessary data to be delivered to the Soft Hub as well as command packets to be delivered to the camera modules from the Soft Hub.

The transport layer comprises a protocol that both the Soft Hub and camera modules use. This is necessary as it defines the order of the interactions between the Soft Hub and camera modules (the protocol logic) and the format of the data (the data format). The protocol is broken into two parts: the data format and the control logic.

The data format must be compact to allow sufficient messages to be transferred each image frame. The communications link must have low latency and this is determined by the protocol and the bandwidth of the physical layer. Latency is used here to describe the time between data becoming available for transmission and it being available to the receiving party or parties.

The communications system must also be reliable to ensure that data sent by the camera modules arrives at the Soft Hub without errors. This can be achieved by error correcting codes in the protocol or by using a reliable communication medium. In this system the

communication medium is assumed to have a negligibly small error rate and error correcting codes are not used. The protocol should be extensible in its ability to transport new types of data with few changes required. Provided sufficient bandwidth is available, the protocol should be human readable to allow the system to be debugged easily using a terminal program. The physical layer and transport layer are described in the next section.

6.1 Physical layer

The physical layer is a three wire serial bus using UARTs as shown in Figure 6.1. From left to right the figure shows the Soft Hub connected via USB to a bridge chip. This chip converts the USB data stream to a standard UART stream. This is shared by the modules connected to the bus. The three bus wires are a transmit line (Tx), a receive line (Rx), and a flow control line (CTS). These provide full duplex communications between the camera modules and the Soft Hub. The UARTs are built into each of the Blackfin DSP chips and into the FT2232C [84] USB bridge chip from Future Technology Devices International (FTDI) Ltd. The bridge chip provides a bandwidth limited to approximately 1 M baud [84] (921600 baud by the PC drivers). Using the common configuration of 8 data bits, no parity bits, and 1 stop bit (8N1), this gives a bandwidth of

$$B_{\mathrm{FTDI}} = \frac{921600}{10} = 90\,\mathrm{K\,bytes/s}. \tag{6.1}$$

The Blackfin UARTs are limited to the system clock divided by 16 [64]. The Black Spot system clock is $S_{CLK} = 107.5\,\mathrm{MHz}$ (see Equation 5.9). Therefore, the maximum baud rate is $S_{CLK}/16 \approx 6.7\,\mathrm{MBaud}$. Using a 8N1 configuration, this gives

$$B_{\mathrm{Blackfin}} = \frac{S_{CLK}}{16 \times 10} = 688\,\mathrm{K\,bytes/s}, \tag{6.2}$$

so a system utilising a Blackfin chip for the Hub could achieve a much higher throughput.

The communications channel is full duplex meaning that the data from the Soft Hub does not share the same path as the data directed to the Soft Hub. Any transmission from the bridge chip is received by all camera modules. Each module may drive the shared data path to the Soft Hub and this makes it necessary to have some way to control which module may use the bus at any given time. This is required to avoid collisions when two modules attempt to use the bus at the same time, otherwise the data that they are transmitting will be lost. In this scheme the camera modules are directed by the Soft Hub when they may use the bus. This is described in more detail in the following bus arbitration section.

Initially a two wire design was used, however, testing revealed that under some circum-

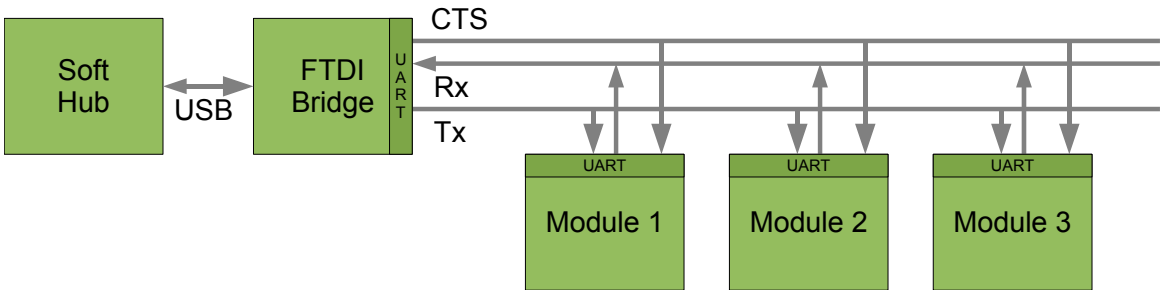


Figure 6.1 Camera modules communicate over a shared bus with the PC.

stances data was lost due to the PC not processing the module data fast enough. A more robust design was favoured using a flow control line (CTS - Clear to Send). This line is asserted by the FTDI chip when the PC cannot service the incoming data buffer fast enough. It is monitored by all the camera modules and when the line is asserted, the module that is currently using the bus ceases to transmit until the line is released by the FTDI chip.

6.1.1 Bus arbitration

The system is designed so that the data path from the camera modules to the Soft Hub is shared. This is achieved by implementing a scheme similar to digital Time Division Multiplexing (TDM) [85]. In this approach time slots are defined for each module within a cycle defined by the camera frame period. A module may only transmit data when its time slot is active. Once a module has finished using the bus the next time slot begins and therefore the time slots do not have a fixed duration (the difference between this approach and TDM). Having variable length time slots is an advantage in that if a module has no data to transmit then it can immediately cease using the bus, freeing it for the next module. However, a disadvantage to this approach is that theoretically a module may take control of the bus for an inappropriately long time, causing other modules to become overloaded with data they wish to send. The control algorithm in the firmware can be written to avoid this situation.

Initial design

Initially, the Soft Hub signaled which time slot was active using a `RequestDataPacket` sent to each module in turn. Each module responded with the data and appended a special packet to indicate that it had finished its transmission. Therefore, each slot involved a communication from the Soft Hub and reply from a Black Spot module. Although this approach works in principle, it proved to be impractical as it relies on a low latency between the Soft Hub and camera modules. Using the FTDI drivers for Microsoft Windows

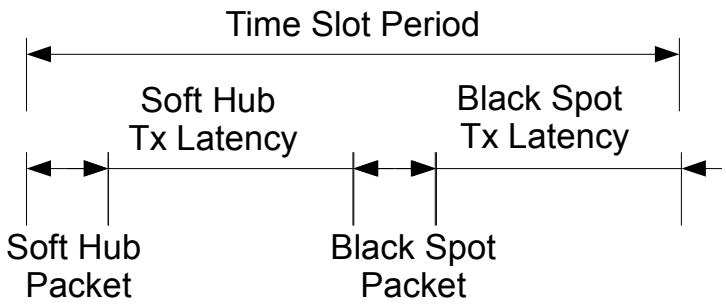


Figure 6.2 In a scheme that requires the Soft Hub to request data from a camera module the time taken to receive the data is the sum of the time it takes to transmit the packet, the latency in the packet transmission, the time it takes for the Black Spot module to send the packet reply, and the latency in the Black Spot reply.

it was difficult to reduce the latency sufficiently. In addition, this approach does not scale well. For example, consider a system with 10 camera modules running at 60 frames per second. This leaves $1/600\,\mathrm{s} = 1.7$ ms per time slot. For there to be time to transmit data in this slot, the round trip latency, i.e., the time to transmit a packet from the Soft Hub and have data returned to the Soft Hub from the camera module must be better than 1.7 ms (Figure 6.2). This proved to be infeasible using the FTDI drivers and the PC hardware and software used to test this system.

Improved design

An alternative approach that was successfully implemented uses a separate bus control signal. A single wire connects each camera module using a general purpose I/O pin as shown in Figure 6.3. The camera modules use this line to signal when they are using the bus and when they have finished. Each camera module is allocated a slot index. At first, time slot number zero is active and the module with slot index 0 sends its data. When the module begins using the bus it sets the bus control signal to a logic one. This alerts all other modules that the bus is busy. When finished the module's timing signal port goes into a high impedance state that effectively disconnects the module from the bus control signal. The bus control line has a pull-down resistor attached that pulls the line down to a logic zero. The falling edge generates an interrupt in each camera module that is registered by a counter in the module's firmware. This counter tracks which slot is current.

At the completion of the time slot (on the falling edge) with index 0, the time slot number 1 becomes active and the module with this slot index takes control of the bus and asserts the bus control signal while it sends its data letting it fall when transmission is completed. The process continues until the last module on the bus has transmitted. At this point the process loops so that after the last time slot has been used time slot zero is active again. This approach works well and is more scalable than the first system. This is due to the

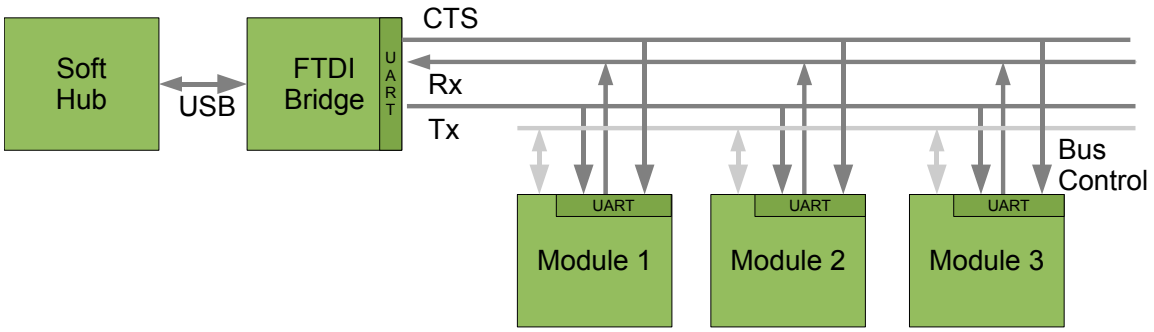


Figure 6.3 The revised communications system uses a dedicated timing line. This line is asserted when the bus is in use.

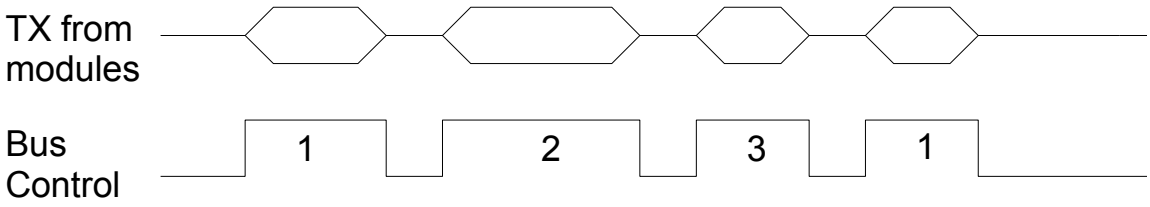


Figure 6.4 A timing diagram showing three modules sending data to the Soft Hub. The bus control signal is being driven by each module in turn. Each numbers represents the module that is controlling the bus, i.e., module number 1 first controls the bus.

low latency associated with detecting the falling edge of the bus control signal. However, one potential problem is that noise on the bus control signal could cause a falling edge to be registered spuriously. In the first method the likelihood of this happening is smaller as only a valid packet can trigger a module to send its data. During testing this has not been a problem. Another problem that affects both approaches is that of the firmware in a camera module crashing causing the module to become unresponsive and the system to stall when the module's time slot becomes active. This can be fixed by allowing the Soft Hub to reset the order of time slots when it does not receive data from the modules after a certain period. However, an attempt to recover may not be the best option as the failure of one or more modules may render the system ineffective. In this case recovery would only be useful if the offending module were reset.

Figure 6.4 shows a timing diagram for three camera modules sharing the bus. The TX signal represents the transmission path from the modules to the Soft Hub. The bus control signal shows the status of the shared bus control signal. The numbers below the signal represent which module is driving the signal. Four time slots are shown. Initially, module 1 asserts the bus control signal and transmits its data. It then enters a high impedance mode allowing the signal to fall. This triggers the second module to raise the bus control signal again and transmit its data. The same process occurs for Module 3 and following the falling edge of the bus control signal Module 1 begins sending data again.

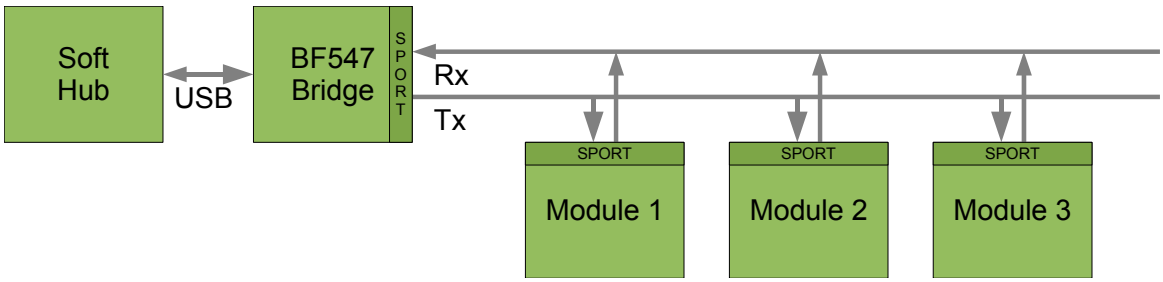


Figure 6.5 A communications interfaces using SPORT peripherals and a Blackfin BF547 DSP is shown. The Blackfin BF547 contains a dedicated USB controller.

6.1.2 Comparison of UART bus and DSP SPORT bus

Although the Blackfin UART peripheral was used in this design, the Blackfin DSP has two synchronous serial ports in addition to the standard UART bus. The UART was chosen for the camera modules due to its simplicity but the synchronous serial ports may be more suitable than the UART in a future design iteration. The Blackfin BF533 [64] datasheet states that the serial port peripherals (SPORTs) support a TDM mode for shared bus access. This would simplify the configuration of the system compared with a shared UART bus and allow the scheme described to be implemented fully in hardware. This would free the firmware from the burden of sharing the bus and free the bus control signal for another use.

The SPORT peripherals are more complex than the UART peripherals and the configuration and running of these modules could increase the complexity of the software and also the hardware. Data must still be delivered to the PC using a USB port or other high speed bus such as a *Firewire* IEEE 1394 bus [86] (at the time of writing USB is the most popular peripheral bus so was the only bus considered). If a SPORT were used then a bridging chip would be required to bridge between the SPORT bus and the USB. Analog Devices make a model of Blackfin processor that contains both a SPORT peripheral and a USB interface. This chip is the ADSP-BF547 [87] and could be used as a dedicated bridging chip between the SPORT interface and the USB (Figure 6.5).

6.2 Transport layer

The transport layer consists of a communications protocol comprising a packet data format and a state machine that defines the states the protocol can have. From an OO perspective, the protocol is a simple mechanism for serialising and deserialising object data. The protocol uses a packet structure in which each packet has a one-to-one mapping between itself and an object in the Soft Hub. A packet consists of three parts. These parts are framing information, header information, and the actual packet data (Figure 6.6). Each part serves

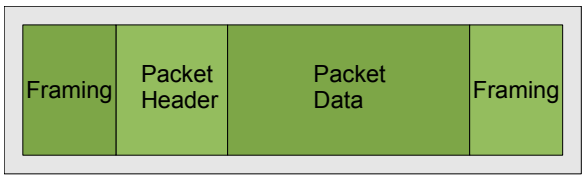


Figure 6.6 A packet consists of a header and data and is surrounded by a packet frame.

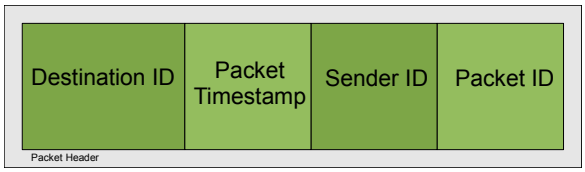


Figure 6.7 The packet header structure consists of the four fields shown.

a different purpose. Firstly, the framing information allows the beginning and end of a packet to be located within a stream of data. This allows a packet decoder to synchronise with the data stream if it begins decoding midway through a packet. This framing information is unique in that it can occur only at the beginning and end of a packet and is not valid data within a packet.

The second part of a packet is the packet header. This contains metadata associated with the packet. For this implementation it is a packet time stamp, the identification of the intended recipient, the identification of the sender, and the packet ID (Figure 6.7). There is latency in the communications system and a packet is not available at the receiver at the same instant that it has been sent. The time stamp ensures that important data such as centroid coordinates can be correctly linked to the time that the centroid was calculated.

Each device using the bus has an ID. This is an integer that is unique to the device. The Soft Hub has the ID '1' and camera modules have IDs greater than one. The '0' ID is used as a broadcast ID and all devices receive packets sent using this ID. The broadcast ID is intended to be used by the Soft Hub only. Finally, the packet ID identifies the type of packet that is being sent. A packet decoder uses this field to determine the packet data. Packets contain different types of data and the length of a given packet is not known until the Packet ID has been determined.

To reduce the computation required by the Black Spot packet decoder, the destination ID is the first field in the packet header. Therefore, the packet decoder need only decode this first field to determine whether it should continue decoding the rest of the packet or ignore it. The packet data varies between packet types but is most often the centroid co-ordinates

<CR> 1,441,2,C,1, 344,304,5690665,5063065,15926,625,25,25,153 <CR>

Figure 6.8 A stream containing a centroid packet is shown. The '<CR>' denotes the carriage return framing character. This character is found at the beginning of a packet and again at the end.

of markers observed by the camera modules.

6.2.1 Protocol implementation

The protocol has been implemented using a text data format. This means that only alphanumeric characters are used for the packet data. This satisfies the desire for human readability (there are not any non-printable characters) but has some disadvantages compared with a binary protocol. In this implementation integers and floating point numbers are encoded as strings of base 10 decimal numbers. The type of packet is encoded as a string also. Each field is separated by a comma and the packet framing begins with a carriage return character and ends with a carriage return. Carriage return characters cannot occur elsewhere in a packet and therefore they are unique to the packet framing. This is advantageous as it means a packet decoder cannot mistakenly believe a packet is beginning or ending when in fact the character occurs in the packet data.

A typical stream containing a centroid packet is shown in Figure 6.8. The packet framing occurs at the beginning and end using a carriage return character (shown by '<CR>'). The next four fields define the packet header. This packet is being sent to the Soft Hub as the destination bus ID is '1'. The time stamp is '441'. The sender ID is '2' which indicates that a camera module with ID '2' sent the packet. The packet ID is given by the 'C' and this denotes that the packet is a centroid packet. Details of all of the packets are given in Appendix F.

6.2.2 Protocol efficiency

Using a text protocol makes it easier for humans to read but it is not as efficient as a binary protocol could be. The consequences of this is that for a given bandwidth fewer packets can be sent using a text protocol when compared with a more efficient binary protocol. The throughput using the text protocol can be estimated approximately. The system's primary purpose is to transfer centroid packets and these constitute the majority of the traffic. The throughput can be calculated based on the length of these packets. Using the bandwidth limit of 90 K bytes/s and a typical centroid packet, the number of centroid packets that can be transferred per second can be approximated. Using a packet length of $L = 55$ (see Figure 6.8), this is $90 \times 2^{10}/55 = 1676$ packets/s. Using a frame rate of $F = 60$ frames/s this is

$$C_{\text{Max FTDI}} = \frac{B_{\text{FTDI}}}{FL} \approx 28\,\text{packets/frame}. \tag{6.3}$$

With each camera module able to track up to 27 markers at one time (see Chapter 4) the bus bandwidth is clearly a limiting factor. The bandwidth of the bus is limited by the FTDI drivers. The theoretical maximum of USB Hi-Speed (defined in the USB 2.0 specification) is 480 M bits/s [88]. If the FTDI drivers were not the limiting factor then the number of centroid packets/frame could be increased to approximately

$$C_{\text{Blackfin}} = \frac{B_{\text{Blackfin}}}{FL} \approx 213\,\text{packets/frame}, \tag{6.4}$$

limited by the Blackfin UART.

An alternative approach to this problem is to improve the efficiency of the protocol. As a text protocol is used, there is a considerable inefficiency. The 26 characters of the alphabet plus the comma and carriage return symbols all can be described with just 5 bits. Presently 8 bits are used to describe this. The same data could be encoded using 5/8 of the bits resulting in a reduction in size of packets of 62.5%. Clearly this is a trivial coding scheme but it illustrates that there is great redundancy in this text scheme.

A binary protocol could be used to improve the efficiency. Using a binary encoding scheme the raw data as it is stored in the Blackfin's memory could be sent directly across the communications link. Packets would then have fixed lengths as opposed to variable lengths. A centroid packet contains four header fields and ten data fields. These are a centroid ID, the coordinates of the upper left corner of the ROI, the X and Y numerators that are used to find the centroid coordinates, the sum of signal pixel intensities (the denominator for the fraction that describes the X and Y centroid coordinates), a count of signal pixels within the ROI, the ROI width and height, and the maximum pixel intensity in the ROI. This is shown in Table 6.1 and the size of each field is given in the fourth column. The sum of these fields is 240 bits or 30 bytes. This binary representation is a reduction in size to $30/55 = 54.5\%$ of the original text packet size that could be used to increase the throughput to approximately 51 centroid packets per frame.

A final approach is to use a compression algorithm to remove redundancy in the data. For example, Lempel-Ziv [85] coding (or a variant) could be used to compress data before it is sent across the communications channel. Using a compression algorithm, the steps for sending data from a Black Spot module would be to create a text or binary centroid packet, encode the packet using a compression algorithm, and send the encoded packet to the hub. The Soft Hub would then decompress the packet and decode it. To illustrate the compression that could be achieved by an algorithm such as this, a stream of data from a Black Spot module was captured. This was stored in a file. The LZMA algorithm (an

Field Name	Field Description	Field Type	Field Size in Bits
Destination ID	The Soft Hub's ID	Integer	8
Timestamp	An integer representing the time that the packet was constructed	Integer	32
Sender ID	The module's ID	Integer	8
Packet ID	A 'C' for centroid	Character	8
Centroid ID	A unique ID for the centroid	Integer	16
ROI X	The X co-ordinate of the upper left corner of the ROI	Integer	16
ROI Y	The Y co-ordinate of the upper left corner of the ROI	Integer	16
X Numerator	The numerator in the fraction that describes the X coordinate, i.e., $X = \frac{\text{X Numerator}}{\text{Pixel Sum}}$	Integer	32
Y Numerator	The numerator in the fraction that describes the Y coordinate, i.e., $Y = \frac{\text{Y Numerator}}{\text{Pixel Sum}}$	Integer	32
Pixel Sum	The sum of pixel values that are above the signal threshold within the ROI	Integer	32
Pixel Count	The number of pixels that are above the signal threshold in the ROI	Integer	16
ROI Width	The ROI width	Integer	8
ROI Height	The ROI height	Integer	8
Max	The maximum pixel intensity of the marker image	Integer	8

Table 6.1 Fields within a centroid packet.

improved version of the Lempel-Ziv algorithm) was used to compress this file. The original file size was 300 Kbytes and the compressed file size was 69 K bytes. This is a reduction in size of $69/300 = 23\%$ of the original size. A compression algorithm that is efficient and light-weight would be required for implementation in the Black Spot firmware. The LZMA algorithm may not be the best algorithm for this but there are many variants that could be tested.

6.2.3 Protocol logic

The second aspect of the protocol is the logic that defines the order that packets are sent in and the states the protocol can take. This is implemented in software using a finite state machine. State machines exist in the Soft Hub and also in the camera modules. The Soft Hub state machine implements the master characteristics of the Soft Hub. It controls when camera modules can use the bus by creating time slots for the camera modules to transmit data in. The camera modules' state machines implement the slave characteristics and prevent the camera modules from using the bus when not permitted to by the Soft Hub.

The state machine in the Soft Hub (Figure 6.9) begins in the uninitialised state. Once the communications port is chosen and opened on the PC, the Soft Hub sends a `RequestData` packet (RQ) to each camera module that it believes is present. Each camera module responds with any data that it has for the Soft Hub and ends by sending a `DataTransferComplete` (TXC) packet. The Soft Hub notes which of the modules responded and then sends a `SetTimeSlot` packet (STDM) to each camera module in the `RequestDataFromOnlineModules` state. This packet informs the module which time slot it must use for transmission. Finally, the Soft Hub sends a `RequestAllData` packet to the broadcast address and this is decoded by each camera module. The first camera module begins transmission and the remaining modules transmit in their time slots.

6.3 Summary

The communications between the Black Spot camera modules and the Soft Hub has been described. The camera modules use a shared UART bus and use a time slot mechanism to coordinate which module can use the bus at any given time. A low latency bus control signal is used to allow the camera modules to signal to each other when they are using the bus. Flow control was also used to prevent data loss at the Soft Hub. The protocol is a text based protocol allowing approximately 28 centroid packets to be sent per frame. This is limited by the bandwidth of the FTDI USB bridge chip. The throughput could be increased by using a binary protocol, compression scheme, or by using a different communications system.

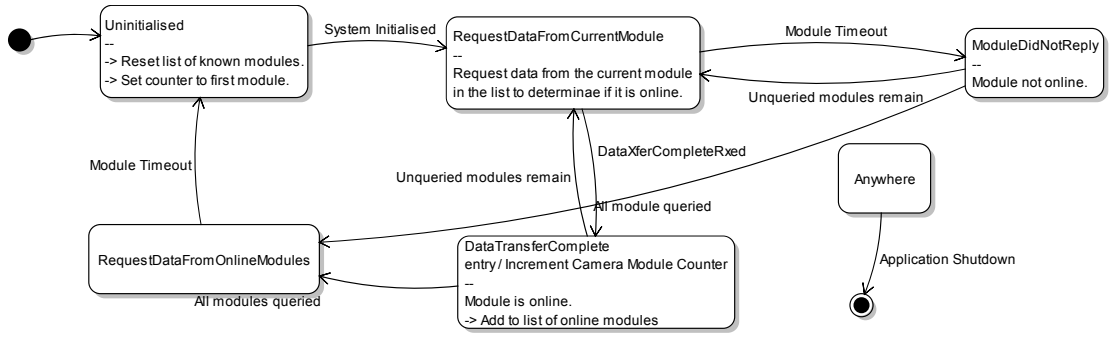


Figure 6.9 The Soft Hub communications state machine. The Soft Hub determines which camera modules are online out of a set of known modules by querying them. If a reply is not received within a timeout period then the module is assumed to be offline. The Soft Hub requests data from all modules that it has determined to be online. If a module fails to send data when it should the process to discover which modules are online repeats.

Chapter 7

Hub Design

The role of the hub is to transform the data presented by the camera modules into an estimate of the pose of the system so that it can be used by other software. The hub consists of two parts; a physical component and a software component. The physical component is an enclosure that holds three camera modules in a rigid fibre glass frame and is referred to as the 'hub enclosure'. These camera modules are positioned so that their FOVs span a larger FOV. The software component is referred to as the 'Soft Hub'. This is software written in C# that runs on a PC. This software is a prototype and the intention is for it to be implemented on an embedded system in the future. In this chapter, the hardware component is described in Section 7.1 followed by the software in the remaining sections.

7.1 Mechanical design

The name Soft Hub reflects that the majority of the hub's functionality is implemented in software. However, there is a physical aspect to the hub and this is the hub enclosure that contains the Black Spot modules. This is constructed from fibre glass and contains three cut-out sections for mounting the modules. The enclosure is designed so that there is a constant interior angle of 150 degrees between the modules providing a slight overlap between the camera modules FOVs. An overlap means markers at the edge of the middle camera's FOV can be observed by the adjacent cameras simultaneously. This creates redundant data and allows for experimentation that would not be possible without this redundancy. For example, this small overlap in FOV essentially provides a small region of stereo vision that could potentially be used for camera self-calibration [89]. Figure 7.1 shows the three Black Spot modules mounted in this housing. This configuration is an initial prototype and the intention is to extend the concept to incorporate more cameras into the hub enclosure in the future.

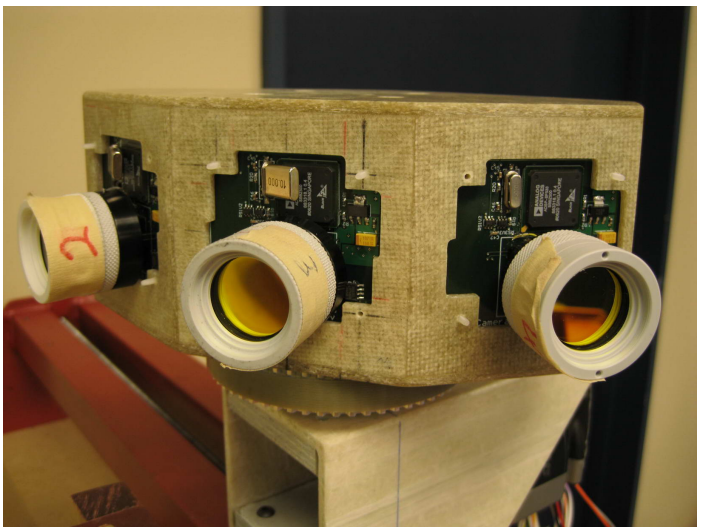


Figure 7.1 The hub enclosure is a rigid fibre glass enclosure than contains three Black Spot modules as shown.

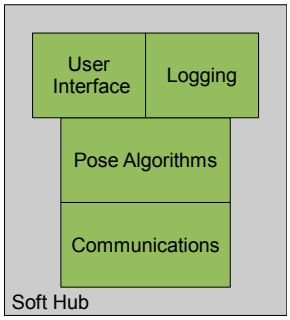


Figure 7.2 The functional blocks of the Soft Hub.

7.2 Soft-Hub design overview

The Soft Hub comprises four major parts (see Figure 7.2). These are the communications system, the pose algorithms, the logging system, and the user interface. The communication system facilitates communications between the Soft Hub and the camera modules in the hub enclosure. In Figure 7.2, pose algorithms refers to all of the classes that relate directly to the problem of pose estimation. These are the classes that are used to produce a pose based on the data from the camera modules. The logging system allows data to be captured and analysed offline. The user interface displays the data available to the system and the output of the pose estimation algorithm. It also allows the user to interact with the system. Finally, logging allows the data from the camera modules and the pose output from the pose estimation algorithm to be written to a MATLAB [90] file.

The four parts of the system are described in more detail in the remainder of the chapter

with emphasis on the pose algorithms as this is the core function of the Soft Hub. The communications system is described in Chapter 6.

To estimate the pose of the hub enclosure the software must do the following:

1. interface to the camera modules and retrieve the centroid data,
2. provide a mechanism to allow data from different modules to be combined,
3. make decisions on the quality of the data from the Black Spot modules and remove poor quality data,
4. use the good quality data coupled with a model of the world to estimate the pose of the hub, and
5. provide a mechanism for logging this pose and an interface to allow this information to be used by other software that may require it.

These details are described in the following paragraphs. Point 1 is the responsibility of the communications layer. Using the protocol described in Chapter 6, centroid data is retrieved from the Black Spot modules. Point 2 is achieved by tagging each centroid with the camera module ID. Using this information it is possible to link the centroid to the coordinate frame of the camera (with respect to the hub enclosure) that the centroid was observed in. Point 3 is not yet implemented but it is intended that the quality measure attached to each centroid is used to determine whether the centroid should be used for pose estimation. Point 5 is described in sections 7.5 and 7.6.

The Soft Hub relies on configuration information to function. This can be grouped into two parts; configuration parameters related to the hub and knowledge about the world. The hub parameters are uncalibrated intrinsic Black Spot parameters, i.e., the focal length of the lenses, the dimensions of the image sensor FOV in pixels, and the size of a pixel in metres. These parameters are modeled internally in the Soft Hub using a Black Spot camera class. The next set of parameters are intrinsic hub enclosure parameters. These define the coordinate frames of each of the three camera modules in the hub enclosure in terms of a hub coordinate frame. In the current software revision, configuration data is hard-coded, however, implementation of a calibration phase in the future should include the ability to accurately estimate this data. The inaccuracies in this data affect the accuracy of the overall pose estimation.

The majority of the data that the hub operates on are centroids from the camera modules. Figure 7.3 shows the flow of this data from the communications layer through the pose

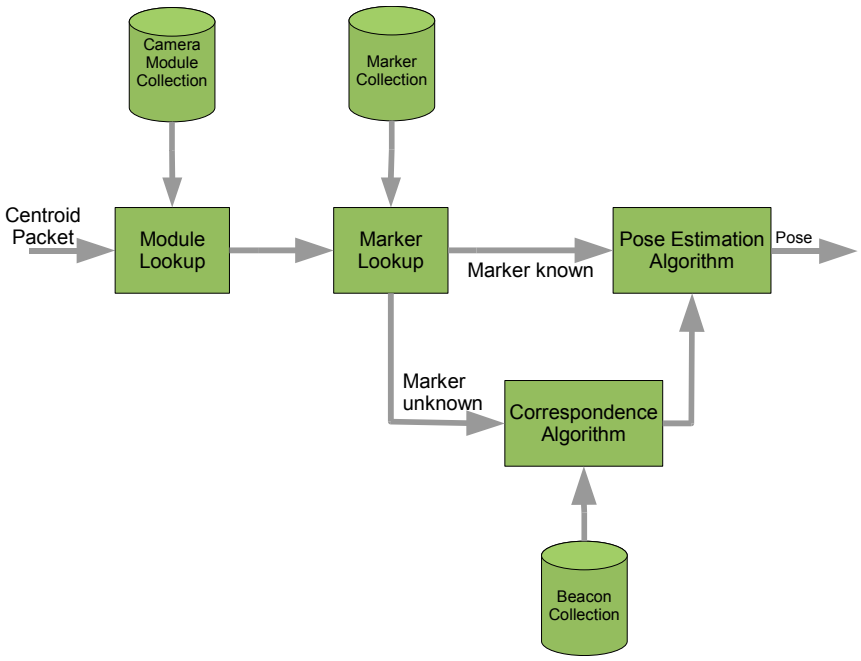


Figure 7.3 This diagram shows the flow of data from the communications layer through the core layer in the Soft Hub.

layer to the user interface and logging system. A centroid packet arrives from the communications layer. The centroid packet is received from one of the camera modules attached to the hub and the module's ID is embedded in the centroid packet. The ID is queried against a database of known modules. The corresponding module is retrieved from the database as well as the pose of the module with respect to the hub enclosure's coordinate frame. The system is configured to know the three camera modules in the hub enclosure, their IDs, and their poses relative to each other.

Next the centroid ID is used as the key to a database of known markers. If the centroid has been linked to a marker in this database then the marker is returned and it is passed to the pose estimation algorithm and used along with other centroids to form a pose estimate. If the centroid is not linked to a marker in the database then another code path is executed. If the pose of the hub enclosure is known then a correspondence algorithm is run that registers the 2D centroid coordinates with the 3D marker coordinates present in the model of the world. These algorithms are discussed in greater detail in the following section.

7.3 Marker registration

The Soft Hub uses a correspondence algorithm to solve the problem of marker registration. This is used to match the images of the markers that the camera modules see to the markers that the system knows exist. This algorithm attempts to match the 2D coordinates of a

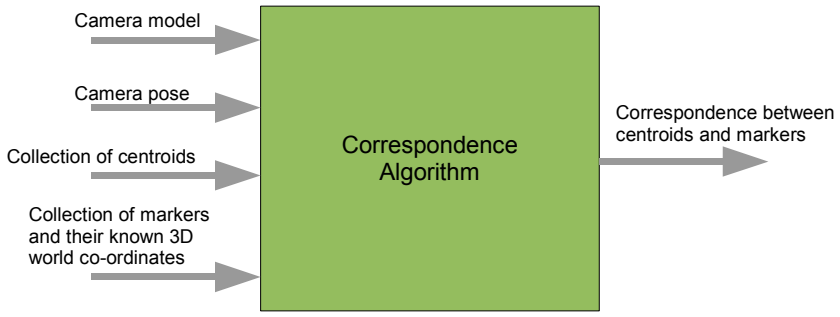


Figure 7.4 The correspondence algorithm matches a 2D centroid with the 3D coordinates of the corresponding marker.

collection of marker centroids with a list of the markers' 3D points. As Figure 7.4 shows, the algorithm requires as input a camera pose, a camera model, a collection of centroids for matching, and a collection of known markers with their 3D coordinates. If the algorithm is successful, the output is the correspondence between each centroid and the 3D coordinates of the marker.

The algorithm uses the pose of the hub enclosure to model the expected 2D coordinates of the markers that would be in view based on the known world data. Views are simulated for the camera modules by using the camera model that is input into the algorithm. Currently a pinhole camera model is used although a more sophisticated model could be used if it were proved necessary. The result of this modeling is a list of 2D coordinates. The problem becomes one of processing the list of 2D coordinates and the measured centroid coordinates to find which centroid from the first list corresponds to which centroid from the second list. In an ideal world, the camera would be perfectly modeled and the camera's pose known exactly. The measured list of centroids would be free of noise and if this were true it would be a case of finding the coordinates of each centroid in the first list which match the coordinates of each centroid in the second list and noting that a match had occurred.

To illustrate this, consider the data in Table 7.1a and 7.1b. Table 7.1a lists three centroids returned by a camera module and their identifiers. Table 7.1b lists the centroids generated by modeling the camera and calculating the 2D positions based on the 3D positions of the three markers. By matching the coordinates of the model points and the measured points the correspondence can be derived as shown in Table 7.1c.

Unfortunately, this example portrays a simplified version of the real world. In the real world measurement noise, uncertainties in the camera pose, and simplifications in the model all cause the coordinates in both lists not to match exactly. A method to find a match between the two lists of centroids given these uncertainties is required. The method devised is unlikely to be optimal but it is easy to implement. The approach has the following steps.

ID	X	Y
1	10	20
2	20	30
3	0	-10

(a)

ID	X	Y
3	20	30
1	0	-10
2	10	20

(b)

Measured ID	Model ID
1	2
2	3
3	1

(c)

Table 7.1 a) Measured centroids in an ideal world. b) Model centroids in an ideal world. c) Correspondence between centroids in an ideal world.

1. For each centroid in the list of measured centroids, do.
2. For each centroid in the list of model centroids, do.
3. Find the Cartesian distance between the model and measured centroids.
4. Repeat from Step 2.
5. If the closest centroid in the list of model centroids is less than a maximum allowable distance then assign a correspondence between the current measured centroid and this model centroid and remove both from the lists.
6. Repeat from Step 1.

After the above steps have been executed for every centroid in the list of measured centroids, and provided the algorithm is successful, then a correspondence between the centroids in both lists is known. The final step is to relate the model centroids back to the 3D coordinates that they were created from. This mapping is known and the final output is a collection of correspondences between 2D centroid coordinates and the corresponding 3D coordinates. The maximum allowable distance in Step 5 is chosen to limit erroneous matches. It is set to a value of a few pixels.

There are a number of things that could cause the algorithm to fail and these are discussed below. Firstly, the algorithm assumes that there is a correspondence between the measured centroids and the 3D marker points. However, this may not always be the case. For example if the camera modules' poses are not known accurately, the model list of centroids may be sufficiently different from the measured centroids such that the algorithm cannot make any matches.

Another possibility is that the algorithm produces a correspondence between the two sets of centroids but that this correspondence is not correct. This could happen for three reasons. Firstly, there may be enough noise and uncertainties in the system to lead the algorithm into wrongly assigning a correspondence between two centroids. This could happen if the noise makes a centroid appear closer than the correct centroid. This may seem unlikely but the problem can be exaggerated if the distance between centroids within either

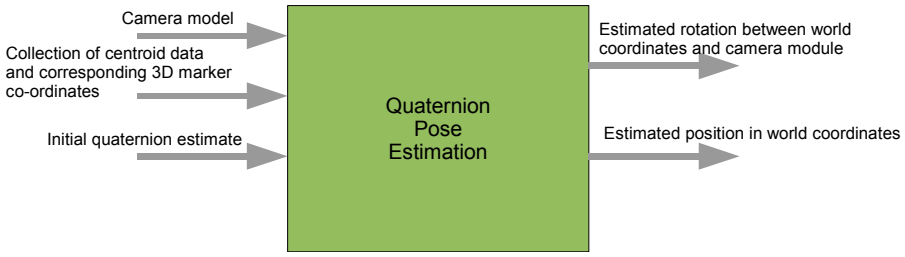


Figure 7.5 The inputs and outputs of the quaternion based pose estimation algorithm are shown in this figure.

list is small. This can occur if the beacon is far away from the camera or if it is viewed by the camera at a large angle off its principal axis. The second reason for an incorrect correspondence is occlusion. The modeled centroids may not match the measured centroid set due to markers being occluded by something. The final possibility is that there may be unknown centroids in the measured list which do not correspond to centroids in the model. This can be caused by reflections and bright points caused by bright lights that are not markers. Due to these problems it may seem that the algorithm has little chance of working but by controlling the above factors the system can be set up so that a correspondence can be found without problems.

7.4 Pose estimation using quaternions

An iterative pose estimation algorithm was designed that seeks to estimate the value of a unit quaternion using the output from the correspondence algorithm. The quaternion describes the rotation between the world coordinate frame and the hub enclosure's frame. The algorithm seeks to minimise a function $F(q)$ using the method of steepest descent [91]. This corresponds to minimising the variance of a cloud of points. The quaternion that results from the minimisation can also be used to calculate the hub enclosure's position. Figure 7.5 shows the inputs and outputs of the algorithm. The algorithm requires as input an initial quaternion rotation, a collection of 3D points and the coordinates of the centroids that correspond to these, and a camera model.

7.4.1 Definition of function to minimise

The following description applies to one camera module. An extension to multiple cameras is given in Section 7.4.4. The goal is to calculate the variance of a cloud of points in 3D described by the function

$$F(q) = \frac{1}{N}\sum_{n=1}^{N} |\boldsymbol{p_n}(\boldsymbol{q}) - \boldsymbol{\mu_p}(\boldsymbol{q})|^2 . \tag{7.1}$$

In this equation N is the number of points, $\boldsymbol{\mu_p}$ is the mean of the points, and the known 3D marker positions and the centroids received from a camera module are used to construct the points $\boldsymbol{p_n(q)}$. F is a function of q, a quaternion rotation, i.e., a quaternion with unit length. The construction of $\boldsymbol{p_n(q)}$ is achieved by constructing a line for each marker that extends from the 3D position of the marker in a direction that depends on the corresponding 2D centroid coordinates. The intersection points of these lines define the point cloud.

The method to construct the lines for intersection is as follows. First, consider the case where the orientation of a camera's coordinate frame matches the world coordinate frame. For example, the camera looks along the Z world coordinate axis such that its X and Y axes match the X and Y axes of the world coordinate frame. Next, for each marker centroid that the module is following, create a vector that points towards the focal point of the camera using the 2D coordinates of the centroid and the focal length of the camera. To explain this further let a centroid have coordinates given by $\boldsymbol{c} = [x, y]$ in pixels. Then the created vector is

$$\boldsymbol{v} = [-xk, -yk, -f]\,, \tag{7.2}$$

in metres where k is the length of a pixel in metres. Construct lines using the marker's 3D location and the direction given by these vectors. The lines will intersect at a single point at the camera's focal point. This intersection point is the camera's position in world coordinates. This is illustrated in Figure 7.6. For convenience of illustration this is shown in 2D. The camera is shown as the blue circle with the centroids shown as red circles. The intersecting lines are shown in green dashes and vectors in blue.

The points $\boldsymbol{p_n}$ are created by finding the intersection point between each line and every other line. If there are M markers then there are M lines. This gives $N = {}^{M}C_2$ combinations of lines combined two ways and the same number of intersection points. The quaternion q in the Equation 7.1 describes the rotation between the world coordinate frame and the camera coordinate frame and is used to rotate each vector. Using the vector $\boldsymbol{v}$ from Equation 7.2, a rotated vector $\boldsymbol{v'}$ is created using the quaternion product

$$q_{v'} = qq_vq^*, \tag{7.3}$$

where q_v is a quaternion with vector part equal to $\boldsymbol{v}$ and $q_{v'}$ is the quaternion whose vector part is equal to the rotated vector $\boldsymbol{v'}$. For this example, $q = (1, 0, 0, 0)$ because the world frame has the same orientation as the camera frame.

Now consider the following scenario. The camera's coordinate system is a rotation and translation of the world coordinate system. The rotation is unknown but it can be expressed by a quaternion. Following the same steps as above, first calculate vectors from

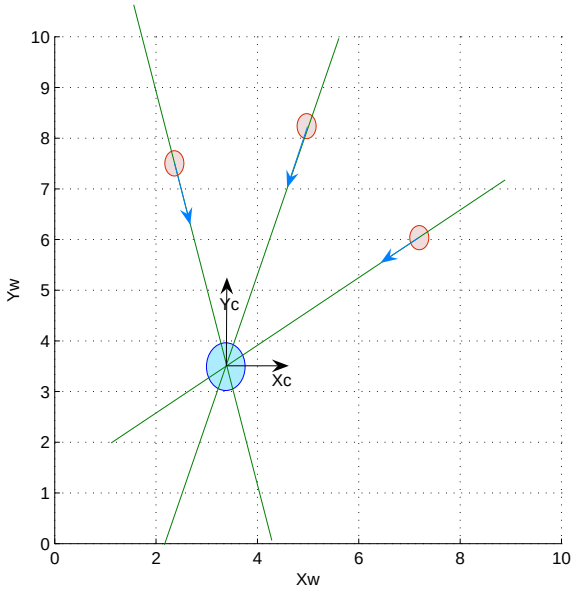


Figure 7.6 With the world axes and camera axes aligned, the intersection of the lines occur at the camera's focal point in world coordinates. In this figure the markers are shown as red circles, the camera as a blue circle, and the vectors pointing towards the camera's focal point are shown as blue arrows extending from the markers. The axes X_c, Y_c represent the camera's axes and are aligned with the world axes X_w, Y_w. The vectors are defined in camera coordinates and the markers' positions are defined in world coordinates. The reason that there is a single intersection point is because the camera and world axes are aligned.

each centroid in the direction of the camera's origin. These vectors are defined in the camera's coordinate space. Now project lines from the markers in world coordinates using each line's corresponding vector. This time the lines will not intersect at one point (Figure 7.7). This is because points and vectors have been used from different coordinate frames. For every pair of lines their intersection point is used to construct the above mentioned point-cloud. The premise is that the size of this cloud is a measure of how close the quaternion q is to the true camera orientation (with respect to the world frame). For example, when the axes are aligned as in the first example the cloud is reduced to a single point.

7.4.2 Extension to 3D

The examples given in Section 7.4.1 have been in 2D. In 2D two lines will always intersect unless they are parallel to each other. In 3D this is not the case. It is unlikely that two lines will intersect. Noise and number precision are two examples of problems that could stop even two specially constructed lines from intersecting at a single point. To overcome this, in 3D, the closest two points on each line are found and these two points are used as the intersection points. This doubles the number of intersection points giving $N = 2 \times^M C_2$ intersection points for M markers[1].

[1]The mean of these two points could be used so that the number of intersection points is not increased

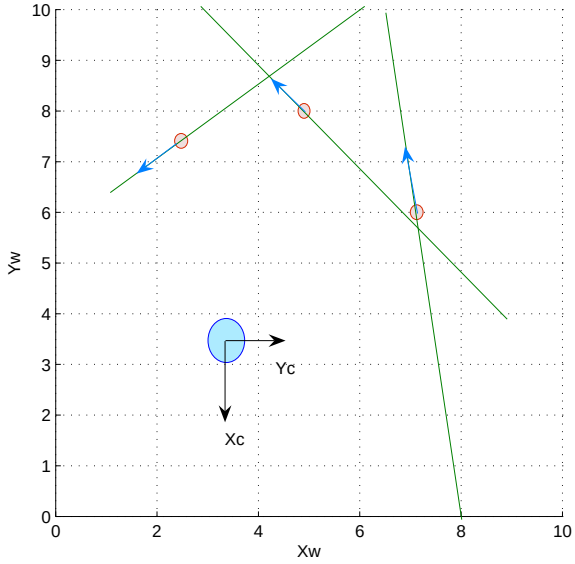


Figure 7.7 Intersection of lines with a clock wise camera rotation of 90 degrees with respect to the world frame.

7.4.3 Minimisation

The function $F(q)$ equals zero when q corresponds to the rotation between the camera coordinate frame and the world coordinate frame. To find a q that minimises $F(q)$ a minimisation algorithm is required. There are many numerical optimisation methods including the method of steepest decent, conjugate gradient decent, Newton's method, the Gauss-Newton method, and the Levenberg-Marquardt method. An overview of these methods is available in, for example, Methods for Non-Linear Least Squares Problems [91]. These methods generally minimise a vector valued function in $\mathbb{R}^n$. This is a problem as $F(q)$ is defined over the unit quaternions that form a 3D manifold in $\mathbb{R}^4$. Schmidt and Niemann propose a method to apply optimisation techniques to a function of unit quaternions by developing a mapping between three rotation parameters and the corresponding unit quaternion [92] . In separate work, Ude [93] derives a method for applying the Gauss-Newton method using unit quaternions.

The method described here modifies the method of steepest descent to allow it to work with unit quaternions. The above algorithms were not used because they were discovered after this algorithm was implemented. The method of steepest descent was chosen due to its ease of implementation and because only first order derivatives of $F(q)$ are used. Evaluation of $F(q)$ is computationally expensive meaning that calculation of first and second order derivatives is also expensive. In the future this minimisation could be replaced with a Gauss-Newton minimisation which has better convergence properties than the method of steepest descent when close to the solution [92]. However, as the Gauss-Newton method uses second order derivatives it should be examined to see whether the improvement in

the speed of convergence is negated by the extra processing required to calculate second order derivatives.

Consider a function $F(\boldsymbol{x})$ that defines an error residual where $\boldsymbol{x}$ is a vector in $\mathbb{R}^n$. The standard steepest descent algorithm begins with an estimate $\boldsymbol{x}_0$ of the minimum of F and the gradient of the function at $\boldsymbol{x_0}$ is calculated. The gradient is a vector of partial derivatives with respect to each of the degrees of freedom of $\boldsymbol{x}$ as given by

$$\nabla F(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial F(\boldsymbol{x})}{\partial x_1} \\ \vdots \\ \frac{\partial F(\boldsymbol{x})}{\partial x_n} \end{bmatrix}. \tag{7.4}$$

The negative of the gradient corresponds to the direction of steepest descent. The steepest descent algorithm moves from the initial estimate in the direction of steepest descent by a step size λ to produce a new value for $\boldsymbol{x}$. This update step is expressed as

$$\boldsymbol{x_{n+1}} = \boldsymbol{x_n} - \lambda \nabla F(\boldsymbol{x_n}). \tag{7.5}$$

Moving in the direction of steepest descent generally yields a better estimate of the minimum of $F(\boldsymbol{x})$. In the case where this is not a better estimate of the minimum, the step size λ is reduced and $\boldsymbol{x_{n+1}}$ is calculated again. This continues until the step size is below a predetermined value and the resulting $\boldsymbol{x}$ is assumed to be the minimum of $F(x)$. Unfortunately, the method may converge to a local minimum rather than the global minimum and this is a good reason for considering another minimisation method in the future.

To minimise $F(q)$ the steepest descent method must be modified to work in the unit quaternion domain. All unit quaternions lie on the unit sphere $\mathbb{S}^3$ in $\mathbb{R}^4$. A new definition of the gradient is required that depends on a parameterisation of $\mathbb{S}^3$. The update step in Equation 7.5 must also be modified to yield a new unit quaternion that corresponds to movement from the previous quaternion in the direction of steepest descent.

First the revised definition of the gradient is defined. The gradient is a vector of small movements in each of the degrees of freedom, i.e., in the original gradient definition these small movements are given by the partial derivatives of $F(x)$ with respect to each degree of freedom. In the new definition the degrees of freedom are defined as small rotations around the three principal axes X, Y, Z expressed as quaternions. The function is evaluated at q and at the three quaternions $q_x q$, $q_y q$, $q_z q$. These quaternions are small rotations around the X, Y, and Z axes measured from q. Using the angle-axis construction , i.e.,

$$q = \cos\frac{\theta}{2} + l\sin\frac{\theta}{2}\boldsymbol{i} + m\sin\frac{\theta}{2}\boldsymbol{j} + n\sin\frac{\theta}{2}\boldsymbol{k}, \tag{7.6}$$

the quaternion q_x is defined as

$$q_x = \cos\frac{\delta\theta}{2} + 1 \times \sin\frac{\delta\theta}{2}\boldsymbol{i} + 0 \times \sin\frac{\delta\theta}{2}\boldsymbol{j} + 0 \times \sin\frac{\delta\theta}{2}\boldsymbol{k} = \cos\frac{\delta\theta}{2} + \sin\frac{\delta\theta}{2}\boldsymbol{i}. \tag{7.7}$$

Similarly, q_y, q_z can be calculated by substituting the Y, and Z axes for (l, m, n), i.e., $(0, 1, 0)$, $(0, 0, 1)$. Here $\delta\theta$ represents a very small angle. This leads to a new definition of the gradient

$$\nabla F(q) = \lim_{\delta\theta \to 0} \begin{bmatrix} \frac{F(q_x q) - F(q)}{\delta\theta} \\ \frac{F(q_y q) - F(q)}{\delta\theta} \\ \frac{F(q_z q) - F(q)}{\delta\theta} \end{bmatrix}. \tag{7.8}$$

Each element is the difference between the function's value moving in one of the three directions and the function at q. The gradient vector is normalised before it is used in the following steps.

A new update step must now be defined that uses the new gradient vector. The update step must produce a new quaternion based on the gradient vector and the step size. The quaternion must represent a move in the direction given by the gradient vector. Each element of the gradient vector $F_1(q), F_2(q), F_3(q)$ represents a proportion to rotate around the X, Y, and Z axes. These can be expressed as angles of rotation around the principal axes given by

$$\begin{aligned} \theta_x &= \lambda\nabla F_1(q), \\ \theta_y &= \lambda\nabla F_2(q), \\ \theta_z &= \lambda\nabla F_3(q). \end{aligned} \tag{7.9}$$

Using the angle-axis construction again, these rotations can be expressed as three new quaternions $q_{\Delta x}$, $q_{\Delta y}$, and $q_{\Delta z}$where

$$q_{\Delta x} = \cos\frac{\theta_x}{2} + 1 \times \sin\frac{\theta_x}{2}\boldsymbol{i} = \cos\frac{\lambda\nabla F_1(q)}{2} + 1 \times \sin\frac{\lambda\nabla F_1(q)}{2}\boldsymbol{i}, \tag{7.10}$$

etc. Now a quaternion corresponding to the overall rotation defined by these three rotations is defined as

$$q_\Delta = q_{\Delta x} q_{\Delta y} q_{\Delta z}. \tag{7.11}$$

The steepest descent algorithm is altered so that it has the form

$$q_{n+1} = q_n q_\Delta. \tag{7.12}$$

This states that the next quaternion is equal to the previous quaternion multiplied by the rotation quaternion q_Δ that rotates q in the direction of steepest descent. The step size λ is reduced whenever moving in the direction of the gradient does not produce a lower value of F. Once the step size is below a threshold, the algorithm stops and the current estimate

is assumed to be the function's minimum. This threshold is set based on experimentation.

7.4.4 Extension to multiple cameras

The description of the algorithm thus far relates to one camera. With more than one camera there will be multiple intersection points when the camera axes are aligned correctly. Each point corresponds to the position of a camera. With multiple cameras, $F(q)$ can be extended to

$$F(q) = \sum_{c=1}^{C} \frac{1}{N_c} \sum_{n_c=1}^{N_c} \left| \boldsymbol{p_{n,c}(q)} - \boldsymbol{\mu_{p,c}(q)} \right|^2 . \tag{7.13}$$

This calculates a combined variance of the intersection points for each camera. In this equation c represents the index of a camera, C represents the number of cameras, $\boldsymbol{p_{n,c}(q)}$ represents the n^{th} intersection point using the c^{th} camera, and $\boldsymbol{\mu_{p,c}}$ is the mean of all of the intersection points from the c^{th} camera. Using this extension the same minimisation process can be used to find a value of q that minimises $F(q)$ and each camera's position can be found in world coordinates by calculating $\boldsymbol{\mu_{p,c}(q)}$.

7.5 Logging system

The logging system takes the output of the pose estimation algorithm and logs this to disk. This enables the data to be analysed in programs such as MATLAB. The intermediate centroid data can also be logged to disk. Logging the centroid data to disk allows the pose to be estimated at a later date.

7.6 User interface

The user interface is a flexible environment that allows the user to visualise the centroids received from the camera modules and displays the pose information. The user interface uses a docking framework that allows windows to be displayed, moved, docked, and closed.

7.7 Implementation details

This section briefly describes the implementation of the Soft Hub in C#. The software is based around the Model View Controller pattern [94]. This is a widely accepted pattern that allows a user interface to be decoupled from the underlying business logic. As the name implies the pattern has three parts. These are the model, the views, and their controllers. The model encompasses the core functionality of the application and is composed of problem domain classes and support classes such as the logging system.

The views implement the user interface. The views are light weight and contain the bare minimum of code. They contain the actual UI components such as buttons, text boxes, and other items. The code that provides logic for the UI but is not part of the application model belongs in the controllers. Controllers provide logic to update views and respond to events such as button clicks. Unit tests are written for controllers but usually not for views as they are typically hard to test.

Chapter 8

Results and Discussion

This chapter assesses the performance of the Black Spot modules. First, the unknowns that are present in the system are identified in Section 8.1. Following this, in Section 8.2, results are given for the modules tested in isolation while stationary. Results gained by testing the pose algorithm using synthetic data are shown in Section 8.3. Due to the performance of the pose estimation algorithm, results from the entire systems are not presented as it was decided that these could be misleading.

8.1 Noise sources

Uncertainties in parts of the system affect the overall performance, therefore it is important to know where these lie. There are two types of unknown; system constants and time varying unknowns. System constants do not change (or change little) during operation. Examples of constant unknowns are the intrinsic camera parameters, i.e., lens focal length, lens distortion, and image sensor pixel size. While it is possible that these parameters may change slightly, for example, due to change in temperature, it is assumed that this change is small and that they are essentially constant. Careful calibration can remove the effects of this type of uncertainty.

There are two types of time varying unknowns; those that can be modeled and those that cannot. An example of a time varying unknown that could be modeled is the variation in measured intensities of the markers such as a 50 Hz sinusoid from the mains power which has been measured on the outputs of the LED drivers. The Black Spot firmware could theoretically compensate for this but, in this case, it would be better to improve the LED drivers.

Some time variant unknowns cannot easily be modeled. Examples are the non-cyclic noise

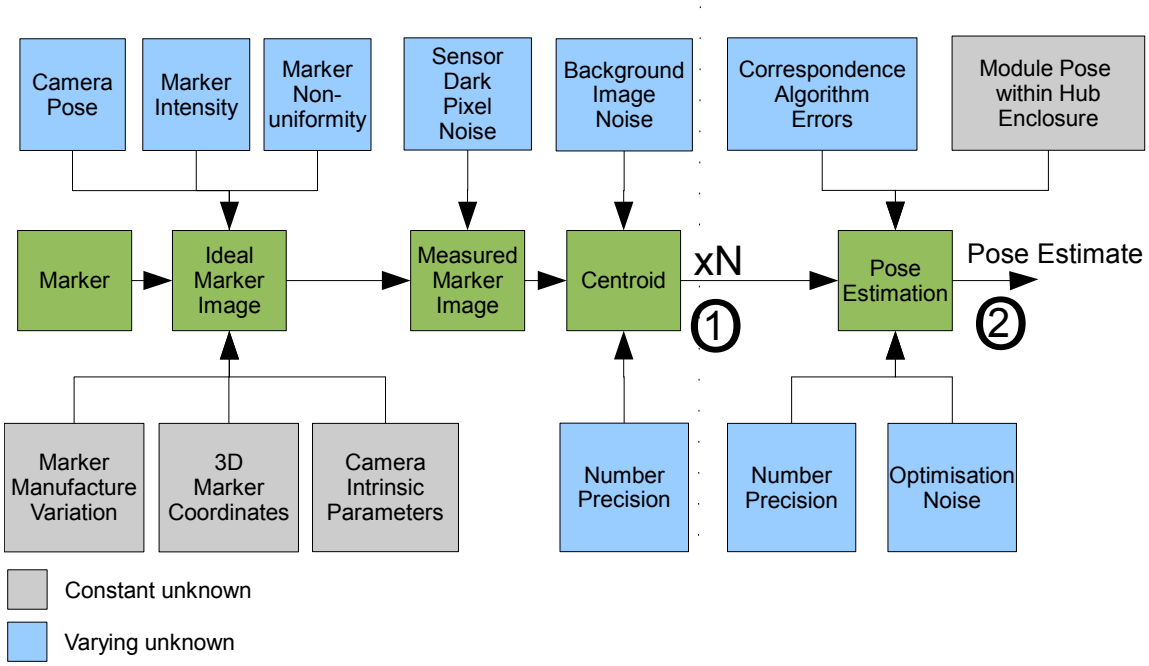


Figure 8.1 The uncertainties affecting different parts of the system are shown. Calibration can reduce the effect of constant unknowns. Modeling can reduce the effect of some varying unknowns. The remainder of the unknowns will reduce the accuracy of the final pose estimate. Point 1 shows the output of a Black Spot module. Point 2 is positioned at the output of the system.

on the markers' current source and the electrical noise on the analog-to-digital converter in the image sensor. This type of unknown cannot be removed by calibration or modeling, and its effect will be seen at the output of the system.

The major constant unknowns are:

- intrinsic camera parameters, i.e. focal length, principal point, pixel size, and sensor skew,
- 3D marker coordinates,
- Black Spot module pose within the hub enclosure, and the
- non-uniformity between markers.

The major varying unknowns are:

- variation in marker intensity due to power source noise,
- variation in marker intensity/marker size due to camera pose,
- variation in centroid position due to background image noise,

- variation in centroid position due to sensor dark noise,
- variation in centroid position due to number precision and variation in pose estimation due to number precision,
- pose algorithm optimisation noise,
- correspondence algorithm errors, and the
- non-uniformity in marker image due to viewing angle and distance.

The points at which these unknowns are introduced into the system are shown in Figure 8.1.

Pose estimation seeks to find only a camera's extrinsic parameters. It assumes that the intrinsic parameters are known and also that precise marker coordinates are available. For the developed pose algorithm (see Section 7.4) this means that, due to uncertainties, there is unlikely to be a value of q that will give $F(q) = 0$. The best performance that can be expected cannot be better than what is dictated by the uncertainties in the constants. The 3D marker coordinates are probably only accurate to ± 3 mm. Only the focal length and pixel size are modeled in the pinhole camera model used so the remaining intrinsic parameters will affect the accuracy further. Finally, the poses of the camera modules within the hub enclosure are also not precisely known. In the experiments in this chapter all LEDs are assumed to be identical, i.e., the variation in the markers due to the manufacturing process is assumed not to have an effect.

Of the varying unknowns, there are at least three that affect the variation in centroid position while the module is stationary. These are the background image noise, sensor dark pixel noise, and variation in marker intensity due to noise on the outputs of the LED drivers. A fourth source of noise is the numerical precision used to store the centroid, however, this is small compared with the other unknowns so is ignored.

Background image noise is caused by the shutter being open sufficiently long so that the captured image contains the background and not just the markers themselves. The majority of this testing has been performed with a shutter period of 300 μs that has, under normal operation, made the background invisible to the sensor. However, at this shutter period image noise may be present in the test environment. To ensure this source of noise does not affect the system, either the system must be used in a similar test environment or a new scheme must be researched to actively suppress the background image.

The second source of noise is referred to as sensor dark noise. This is the noise that would be recorded if the image sensor captured data in a dark environment. This noise is a combination of a number of types of noise including thermal noise and noise introduced during

the analog to digital conversion stage inside the sensor [95]. The centroid calculation determines that pixels below a certain threshold are noise and discards them. Testing has shown that using a pixel value of 4 as a threshold in the centroid calculation works well.

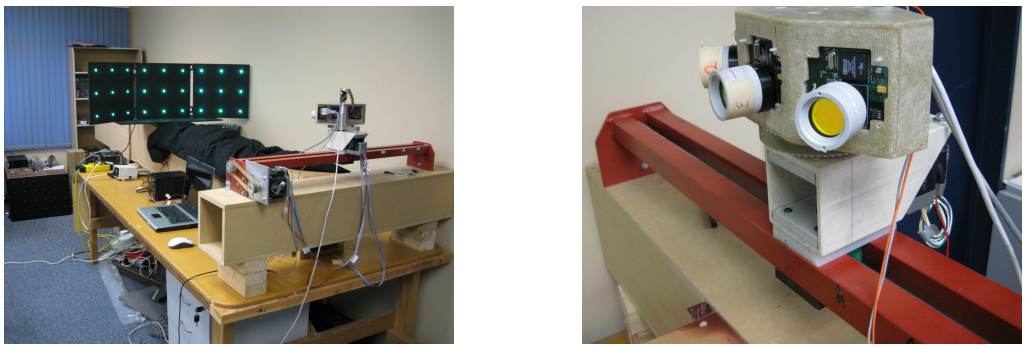
The third source of noise is due to the noise on the LED drivers. Ripple from the mains supply introduces a 50 Hz oscillation in pixel intensity. As is shown later this does not affect markers with large intensities as much as markers with lower intensities.

There are also unknowns in the pose estimation stage. Again number precision is a source of noise although negligible. If the correspondence algorithm links a centroid to the incorrect marker then this could introduce a large uncertainty, however, this is a problem that can be removed entirely by the successful application of the correspondence algorithm. The last unknown relates to uncertainties that could be introduced by the pose algorithm itself. In the algorithm described in Section 7.4, the minimisation terminates when the cloud size is below a specified value or the step size is reduced to below a specified pre-set. Setting the required cloud size directly affects the maximum performance that could be expected with this algorithm. Ideally the cloud size should be set low enough so that the uncertainties due to the pose algorithm are lower than the uncertainties introduced by the variation in centroid positions.

8.2 Black Spot module testing (2D testing)

In this section the Black Spot module test results are presented. By definition the performance of the markers are also tested as testing can only occur with both marker and camera. This section discusses tests up to Point 1 in Figure 8.1. The main output from the modules is a stream of centroids. The variation in the centroid positions is measured while the Black Spot remains stationary. This variation is used to assess how the uncertainties introduced in Section 8.1 affect the performance. The approach taken is to isolate each variable and determine what effect it has on the overall centroid variation. It is assumed that the variables are linearly independent allowing each unknown to be treated in isolation. The parameters that can easily be controlled are the marker intensity, the observed marker image size, and the marker image position in the FOV. All three depend on the camera and marker poses.

By fixing the poses, the effect of marker intensity on centroid variation can be determined (Section 8.2.2). Marker image size is isolated in Section 8.2.4. The marker image position is tested in Section 8.2.3. In Section 8.2.5, the variation in centroid position over a large period is tested. This tests not only the module but the entire test setup as any expansion or contraction due to variations in temperature of the equipment will cause the centroid positions to move. The performance between modules is tested in Section 8.2.6.



(a) (b)

Figure 8.2 (a)The testing configuration consists of two computer controlled translation rails that alter the distance between the hub enclosure and the testing beacon. The long translation rail allows the beacon to be moved towards the hub enclosure. (b) The hub enclosure is attached to the shorter rail and a rotational turret allows the angle of the hub enclosure to be varied by a computer.

During stationary testing, measurements were made using the configuration shown in Figure 8.2a that consists of a beacon, two computer controlled linear rails, a rotational turret, and the hub. The beacon comprises three panels of 3 by 3 LEDs and is shown at the end of the longer of the two rails. The markers are placed in a regular pattern with a distance of 0.15m between adjacent markers. The angles between the two outer panels and the centre panel can be adjusted.

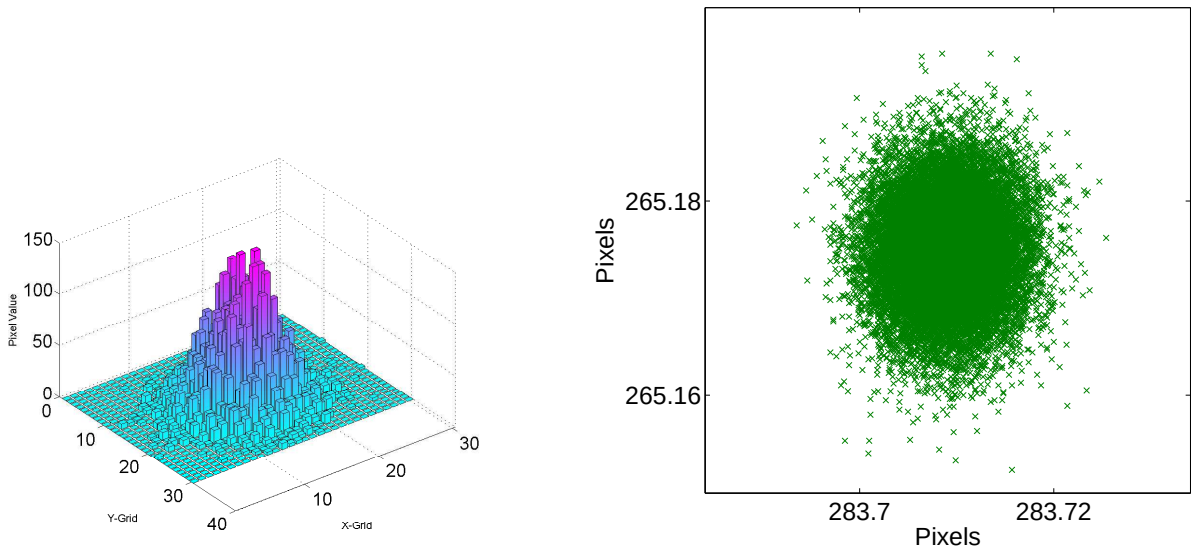
The two rails are oriented perpendicular to one another to provide a range of testing configurations. The long rail allows the beacon to be translated towards the hub enclosure. This is achieved by way of a stepper motor that allows precise movements to be made. The shorter rail translates the hub enclosure in a direction perpendicular to the long rail and is connected to another stepper motor. The hub sits on a turret whose angle can be adjusted by a third motor (Figure 8.2b).

The measurements discussed in the following subsections were taken at fixed distances. This distance refers to the distance between the test beacon and the centre of the hub enclosure, measured with a tolerance within ± 1 cm.

8.2.1 Measure of centroid variation

The standard deviation of an ensemble of centroid coordinates is used as a measure of spread throughout these results. Figure 8.3a shows a plot of the distribution of centroids captured from a module. The plot has a Gaussian shape. Gaussian statistics are assumed for the centroid X and Y positions for the remainder of this chapter.

Figure 8.3b shows a scatter plot of centroids. The axes cover a 0.05 by 0.05 pixel region



(a) (b)

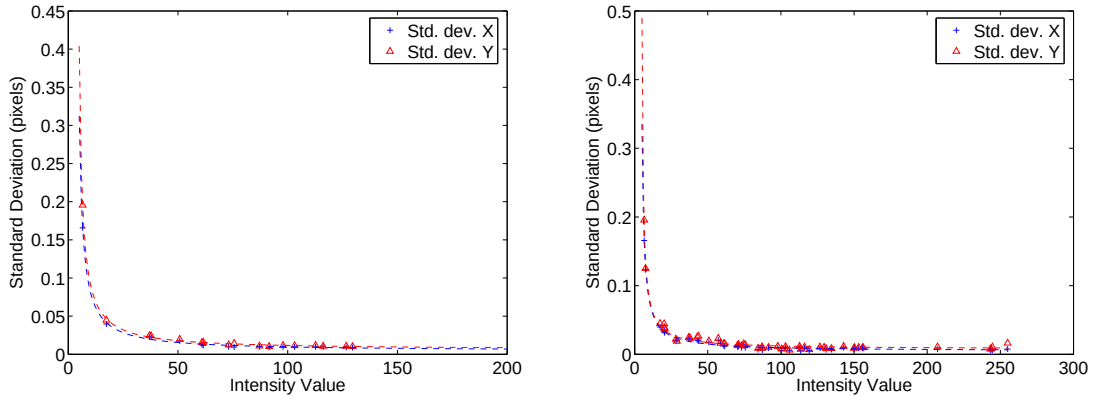
Figure 8.3 (a)The shape of a 3D histogram plot of centroid positions indicates that the X and Y co-ordinates of the centroids are approximately Gaussian distributed. (b) The spread of centroids are shown falling within a 0.05 by 0.05 pixel region.

demonstrating how small the variation in position can be.

8.2.2 Variation due to marker intensity

Clarke et al note that "*The standard deviation of a centroid co-ordinate is approximately inversely proportional to the maximum intensity level.*" [67]. This relationship is tested in this section. It should be noted that the observed intensity is dependent on the shutter period of the camera which was set to 300 μs in this test. Three markers were fixed near the centre of a Black Spot module's FOV. The markers' intensities were varied in discrete steps while the centroids of the marker were recorded over time. The data set was analysed and the spread of the centroids over time were calculated for each intensity. The resulting graph in Figure 8.4a shows the effect of intensity on the spread of the centroids for 1 marker. As predicted by Clarke et al. [67], the graph has a hyperbolic shape.

The data were analysed again using all three LEDs and plotted in Figure 8.4b. This produced a plot with the same shape and a similar curve was fitted to the data. As shown, increasing the maximum intensity value of a marker above a value of approximately 80 makes little improvement to the spread of centroids. Consequently, a lower weighting should be given to makers of intensity less than this value. Shortis et al. 1994 [47] note that in their simulations, the location accuracy of the centroid calculation degrades with an increase in saturation of the image. In this case, saturation is present when the maximum



(a) (b)

Figure 8.4 The spread of the centroids is approximately inversely proportional to the maximum intensity of a marker. (a) Data collected using one marker. (b) Data collected using three markers. Both plots have the same shape and curves were fitted to each of them. There is little improvement in centroid variation for markers with intensity greater than approximately 80 for a shutter period of 300 μs.

intensity is 255. It is concluded that markers with intensities between 80 and below 255 can be treated equally.

The fitted curves have the form

$$\sigma = \frac{p_1 I + p_2}{I + q_1}, \tag{8.1}$$

where I is the intensity, p_1, p_2, q_1 are constants and σ is the standard deviation. For the plot in Figure 8.4b, the fit for the standard deviation in X is

$$\sigma_{\mathrm{X}}(I) = \frac{4.65 \times 10^{-3} I + 0.468}{I - 3.56}, \tag{8.2}$$

and the fit for the standard deviation in Y is

$$\sigma_{\mathrm{Y}}(I) = \frac{7.41 \times 10^{-3} I + 0.417}{I - 4.07}. \tag{8.3}$$

8.2.3 Variation due to viewing angle

This section attempts to determine whether there is a dependence between the variation in centroid positions and the position of a marker image in the camera's FOV. To test this the rotation turret was used to vary the camera angle. This varied the positions of the visible markers in the camera's FOV. Rotating the camera corresponds to a change in θ_1 as defined in Figure 8.5. In this test it is assumed θ_2 does not affect the variation in centroids measured from this marker, rather that this angle only affects the apparent intensity of the

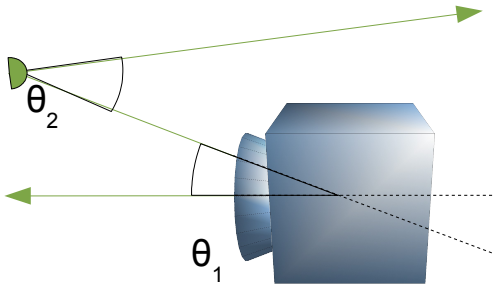
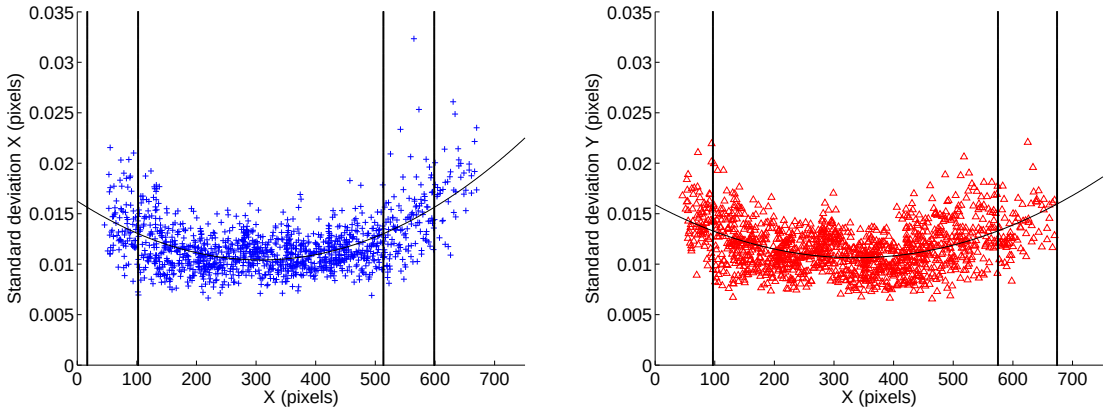


Figure 8.5 There are two angles that could have an effect on the standard deviation of the centroids from a marker. The first is the angle between the camera's principal axis and a line that extends from the camera's principal point to the marker (θ_1). The second is the angle between this line and the marker's principal axis (θ_2). The marker's principal axis is shown by the top green arrow and the camera's principal axis is shown using the bottom green arrow.



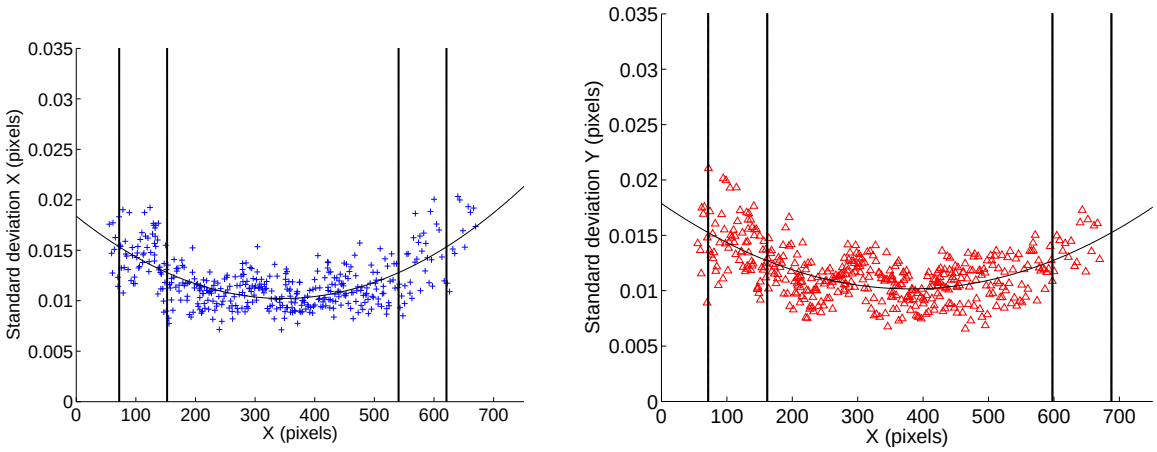
(a) (b)

Figure 8.6 These graphs plot the centroid variation versus X position. a) shows the standard deviation in X and b) shows the standard deviation in Y.

marker and the number of visible marker pixels. For the analysis all measured marker intensities were above 80 meaning that the change in intensity due to a change in θ_2 has a negligible affect on the results (see Section 8.2.2).

Figures 8.6a, b show plots of centroid standard deviations in X and Y versus X coordinate. The graph appears to have a flatter region in the middle of the FOV and a region at either side of the FOV where the standard deviation increases. In these plots, data were used from every marker in the FOV. Quadratic curves were fitted to the plots and vertical lines were added showing where the fitted standard deviation curve is 25% and 50% higher than the minimum.

The same data set was analysed but this time markers from approximately the same Y



(a) (b)

Figure 8.7 These graphs plot the centroid variation versus X position for a band of markers with similar Y values. a) shows the standard deviation in X and b) shows the standard deviation in Y.

position were used. This was an attempt to remove any bias that could be caused by using a varying Y position. Figure 8.7 shows the outcome. The graphs have the same shape and similar quadratic curves provide a good fit to the data. A better test would be to use data from just one marker as it is translated across the entire FOV and is suggested as future work.

A stationary measurement was taken and the mean marker position was plotted with error bars (Figure 8.8). The error bars represent the standard deviation of the marker position and are increased 1000$\times$ so that they can be seen on this graph. This provides an overview of the parts of the FOV that appear to give better results. The six markers on the right have larger standard deviations than the markers in the middle. This agrees with the results in Figures 8.6a,b as these markers are near the outside of the FOV. The markers in red with a square box are those that have intensities below 80. The large standard deviations in these markers could be due to the low intensity so should not be considered. A similar data set that had a larger range in Y values was analysed and the results support that shown in Figure 8.8.

Using these results it is concluded that good results can be obtained by using markers in a region between approximately $100 < \mathrm{X} < 500$ based on Figures 8.6a, b. In this region all standard deviations are within 25% of the minimum. As the rail was translated in the X direction it was difficult to find a trend in the Y direction. The quadratic curves fitted to these figures are

$$\sigma_{\mathrm{X}}(\mathrm{X}) = 61.6 \times 10^{-9}\mathrm{X}^2 - 37.9 \times 10^{-6}\mathrm{X} + 16.3 \times 10^{-3}, \tag{8.4}$$

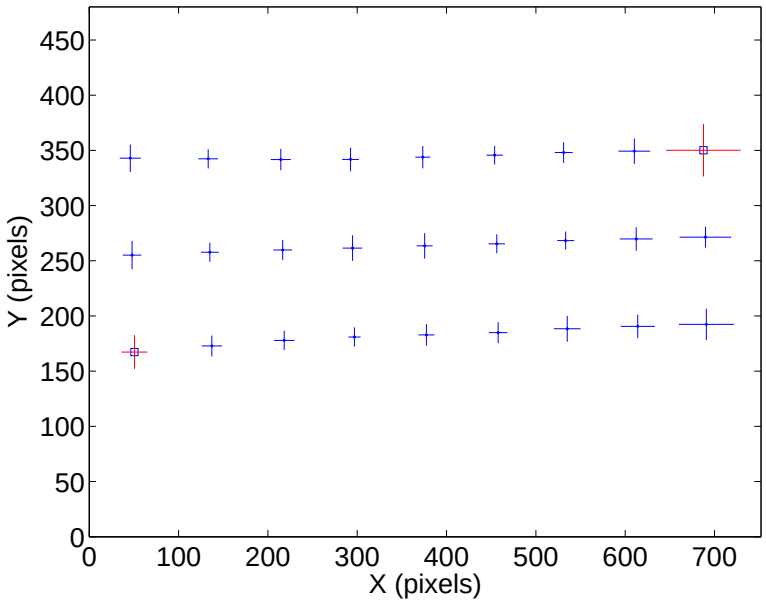


Figure 8.8 The positions in the camera's FOV of marker are shown with error bars representing the spread in X and Y magnified by 1000. It appears that the markers near the edge of the FOV have slightly higher standard deviations. This is consistent with the results in Figures 8.6a, b.

$$\sigma_{\mathrm{Y}}(\mathrm{X}) = 46.5 \times 10^{-9}\mathrm{X}^2 - 31.3 \times 10^{-6}\mathrm{X} + 15.9 \times 10^{-3}. \tag{8.5}$$

The data were also analysed in terms of distance from the centre of the FOV. Despite the fact that the centre of the FOV is not precisely known, it appears that a similar trend exists as shown for a varying X value but in this case it is for a varying radial distance.

8.2.4 Variation due to distance

The Black Spot modules are designed to track markers in their FOV at a distance of between 0.5 m and 3 m. Therefore, it is important to test the modules' performance between these distances. A Black Spot module is fixed at distances of 0.5, 1, 1.5, 2, 2.5, and 3 m with respect to the test beacon, facing the markers directly. The standard deviations of the centroids over 2 minute periods were calculated at the different distances. Table 8.1 shows the results obtained. The mean of the centroid standard deviation of each marker is measured. This is given along with a measure of spread from the mean as an indicator of how reliable the mean is.

As shown in Table 8.1, the spreads in standard deviation for the 1 m and 0.5 m measurements are significantly higher than the rest. The data for these distances were analysed and it was determined that one marker was producing much higher centroid variations than other markers at this distance. This marker had a low maximum intensity (significantly

Distance (m)	Mean std. dev. X	2σ value	Mean std. dev. Y	2σ value
3.0000	0.0131	0.0037	0.0143	0.0040
2.5000	0.0104	0.0031	0.0119	0.0042
2.0000	0.0111	0.0040	0.0113	0.0036
1.5000	0.0092	0.0042	0.0096	0.0025
1.0000	0.0137 / 0.0079	0.0197 / 0.0040	0.0116 / 0.0089	0.0087 / 0.0007
0.5000	0.0521 / 0.0170	0.0711 / 0.0134	0.0684 / 0.0114	0.1139 / 0.0012

Table 8.1 The mean spread in centroid positions appears to decrease as the module nears the camera until a distance of 1 m. At 0.5 m the spread increases again. The values to the left of the '/' in the last two rows were calculated before outliers were removed. The values to the right have had the outliers removed.

less than 80) and was removed from the data set and the table recalculated. The recalculated values are shown following the '/' in the table. One possible explanation for this low intensity spot is a reflection of the rail, as observed on a number of occasions while collecting data. This shows that there is a need for the system to be able to remove poor quality data.

The results have a tighter spread with these outliers removed, however, the mean standard deviation in X at 0.5 m is still large. Using the adjusted results it appears that the mean spread in marker centroids decreases as the module nears the camera but then begins to increase again somewhere between 1 m and 0.5 m. At 0.5 m there are only three markers visible (after removing the outlier). The first marker has a higher standard deviation than the other two and appears to be lifting the mean standard deviation. The intensity of this marker at 0.5 m was investigated revealing that the marker was saturated meaning some of the marker pixels were at a value of 255. This could explain the lower performance.

Interestingly, the standard deviation in the Y direction is often slightly greater than the standard deviation in the X direction. This is consistent throughout the measurements. As yet it is not known what causes this and this is an area for future research. However, as this difference is small the performance of the system will not be affected.

Another data set was collected in order to produce a more detailed trend. The beacon was translated towards the Black Spot in 0.06 m increments starting at a distance of approximately 3 m and ending at approximately 0.5 m. A plot was produced of the distance versus the standard deviation in the X and Y directions comprising 40 points as shown in Figure 8.9. In Figure 8.9 the mean standard deviations of all visible markers are plotted. Outliers have been removed based on the markers intensity values. As the beacon moved towards the Black Spot, some markers disappeared from view and therefore the mean was calculated with a varying number of markers. The last data point was discarded as it was

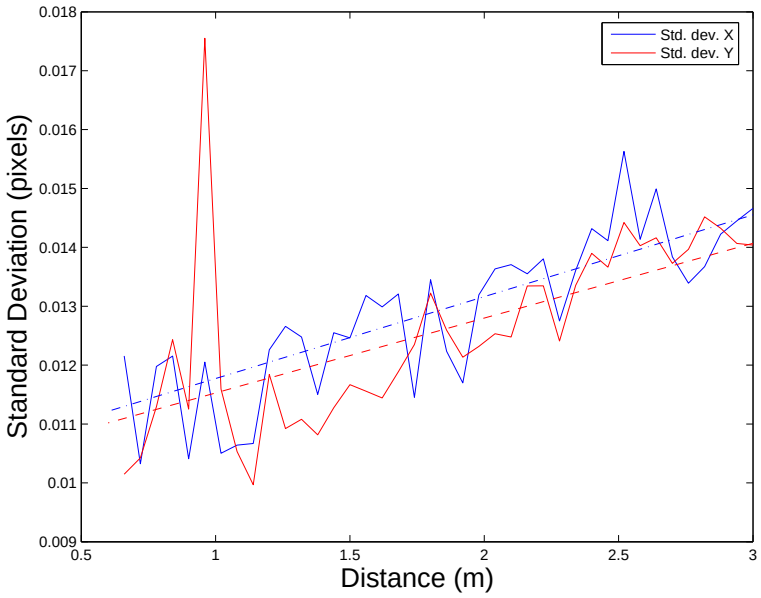


Figure 8.9 The centroid variation increases approximately linearly with distance. The mean standard deviation of all visible markers was used in this plot. Outliers were removed.

generated from the mean of only two markers and is considered to poorly represent the general performance of the system. The standard deviations are approximately linearly proportional to the distance between 0.6 m and 3 m as shown by the two trend lines. The spike observed at 1 m is considered anomalous, resulting from an external disturbance of the measurement (see Appendix G).

The trend between 0.6 m and 3 m for the standard deviation given in Figure 8.9b in X is

$$\sigma_{\mathrm{X}}(D) = 1.40 \times 10^{-3} D + 10.4 \times 10^{-3}, \tag{8.6}$$

and the standard deviation in Y is

$$\sigma_{\mathrm{Y}}(D) = 1.30 \times 10^{-3} D + 10.3 \times 10^{-3}. \tag{8.7}$$

8.2.5 Long term stability

Factors that could affect stability include temperature and ambient lighting. To determine whether there is any drift in the centroid data over time a long term measurement was made using each of the three Black Spot modules. Data were collected for approximately 8 hours throughout a working day. Each module was stationary and faced the stationary test beacon. Data from each module were not captured concurrently so cannot be compared directly but the conditions are believed to be sufficiently similar to make the statistics broadly

comparable. These large data sets show how the centroids vary with the changes that can be found in an office environment during a typical day.

It should be noted how drift could affect the performance of the system. First, assume that the system is calibrated only at start up and therefore any drift throughout the day would degrade the performance of the system. A large drift in the measured centroid positions from calibration would lead to a large degradation in performance. Therefore it is important that any drift measured is cyclic and that the maximum magnitude of the drift is small.

Each data set was captured using a shutter period of 300 μs with a 2 m separation between camera and beacon. In each of the three data sets, a marker near the centre of the FOV was used, as these provide the best performance as discussed in Section 8.2.3. The minimum standard deviations of each of the centroids over the entire data set was also calculated in X and Y. Results are shown in Table 8.2. The lowest standard deviation over the entire data set was from Module 4[1] and this shows that the centre marker has a standard deviation of 0.0141 pixels in X and 0.0234 pixels in Y. The standard deviations in pixels can be converted to metres as the focal length of the lens and distance between the LEDs and the camera module are known. The lowest standard deviation corresponds to 28.2 μm in X and 46.8 μm in Y at 2 m. The worst result was from Module 2 and shows a standard deviation of 0.0507 pixels in X and 0.0842 pixels in Y corresponding to 101.5 μm in X and 168.4 μm in Y. These values indicate that even with only one calibration period at the beginning of a 7 hour shift (and assuming the data set is representative), the 2D centroid variation is low.

In the future, the intention is that the system will continuously recalibrate itself and it is assumed that any drift can be accounted during calibration. Assuming that the system re-calibrates every 2 minutes. Using the data sets available this corresponds to 100 samples of data. For each sample, the standard deviation of the 100 data points surrounding this sample was calculated. This produced a matrix of standard deviations. The columns represent the centroid number and the rows represent time. Therefore, the element at the m^{th} row and n^{th} column represents the standard deviation of the centroids from the n^{th} marker. This standard deviation is calculated using the 100 data points surrounding the m^{th} instant. This was done for each data module.

The graph in Figure 8.10 shows an example of this data showing the standard deviation as it varies over time for Module 4. The values shown are the mean of the standard deviations of the markers and are calculated by taking the mean of each row of the matrix of data. The standard deviations shown are of the centroids' Y coordinates. The graph shows that the mean standard deviation is generally small but there are a number of large spikes in the

[1]Modules are numbered corresponding to their bus ID. Hence, they are numbered 2, 3, and 4.

	Module 2 Value (pixels)	Module 3 Value (pixels)	Module 4 Value (pixels)
Worst std. dev. in X of centre marker	0.0507	0.0457	0.0312
Worst std. dev. in Y of centre marker	0.0842	0.0657	0.0416
Mean std. dev. in X of centre marker	0.0281	0.0303	0.0200
Mean std. dev. in Y of centre marker	0.0681	0.0601	0.0337
Best std. dev. in X of centre marker	0.0212	0.0239	0.0141
Best std. dev. in Y of centre marker	0.0459	0.0535	0.0234

Table 8.2 The best, mean, and worst standard deviations for the centre markers over a period of approximately 8 hours are given in this table.

	Module 2 Value (pixels)	Module 3 Value (pixels)	Module 4 Value (pixels)
Worst std. dev. in X of centre marker	0.0103	0.0144	0.0095
Worst std. dev. in Y of centre marker	0.0135	0.0164	0.0240
Mean std. dev. in X of centre marker	0.0081	0.0092	0.0061
Mean std. dev. in Y of centre marker	0.0086	0.0114	0.0074
Best std. dev. in X of centre marker	0.0062	0.0066	0.0047
Best std. dev. in Y of centre marker	0.0059	0.0078	0.0045

Table 8.3 The data in this table represents the short term stationary performance of the modules. The best, mean, and worst noise performances over 2 minutes are given.

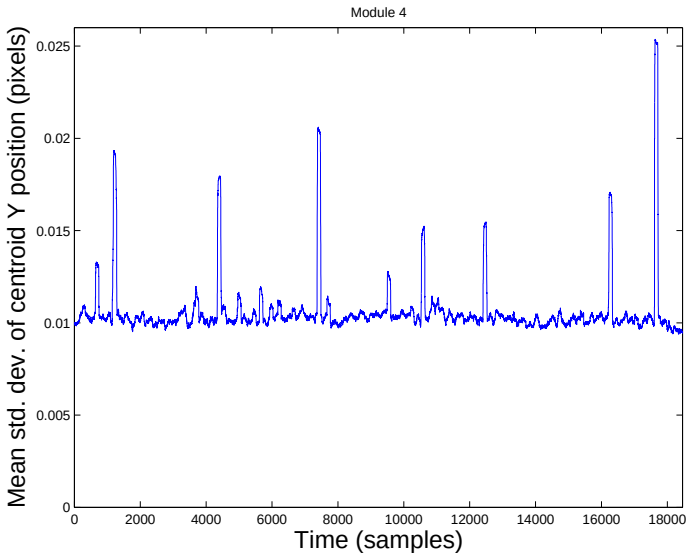


Figure 8.10 The standard deviations of the centroid's Y position over 2 minute periods throughout a day. The spikes are outliers as discussed in Appendix G.

graph. These spikes are treated as outliers and their cause is investigated in Appendix G.

These outliers should not have a large effect on results due to the duration of the spikes with the exception of the worst case results shown in Table 8.3.

The two minute intervals in this data set can be analysed to produce a best case, worst case, and mean case scenario. Again, the marker closest to the centre of each module's FOV was chosen for this analysis. The best case scenario was calculated by looking at the smallest standard deviation across all 2 minute windows. The mean case was calculated by taking the mean of the standard deviations over time and the worst case was calculated by finding the largest standard deviation. The results are shown in Table 8.3. The figures in the rows named 'Best std. dev. of centre marker' are the best noise performance that one could hope to expect over a 2 minute period. Module 4 has the best performance in the X direction with a standard deviation of 0.0047 pixels and 0.0045 in the Y direction. These numbers correspond to a standard deviation (at the measurement distance of approximately 2 m) of 9.3 μm and 9.0 μm respectively. The worst case is the largest standard deviation of the centre marker over all 2 minute windows. This is shown in the Table 8.3 in the rows named 'Worst std. dev. of centre marker'. These values are distorted by the outliers as shown in Figure 8.10 and, until further tests can be completed, these results have little significance. The mean standard deviation is more informative and has values of 0.0061 pixels in X and 0.0074 pixels in Y. This corresponds to 12.2 μm and 14.8 μm respectively.

In summary, the system is stable over the long term. Without continuous calibration the best result is a standard deviation of 1.4% of a pixel. Using a threshold of 2 standard

deviations and assuming Gaussian statistics, this corresponds to an X resolution at 2 m of $\pm 56.4\,\mu\mathrm{m}$ with 95% confidence. If the system is calibrated every 2 minutes then using the same 95% confidence interval this corresponds to a X resolution at 2 m of at best $\pm 18.6\,\mu\mathrm{m}$. Based on this, the system could operate without continuous calibration but it would be more favourable for the system to recalibrate often to obtain the best noise performance.

8.2.6 Comparison between modules

Module 2 and Module 4 were constructed using the MT9V023 and Module 3 was constructed using the MT9V022, an older version of the sensor. Statistically it is not possible to draw conclusions about the performance between the two image sensors as the sample size is too small. However, the long term data shows that all three modules have similar performance. The best performance for the two minute interval data in the Y direction ranges between 0.0045 pixels for Module 4 and 0.0078 pixels for Module 3. As seen in Table 8.3, Module 3 usually has poorer performance than the other two modules but without a larger sample size it cannot be concluded with certainty that this is because of the older image sensor.

8.2.7 Background noise

Background noise can have a large influence on the centroids. Figure 8.11 was obtained during testing. This measurement contains data from one marker but two groups of centroids can be seen. This can be explained by the position of the ROI that surrounds the marker. Although the marker is stationary, the ROI moves throughout the measurement. The explanation for this is that noise on the marker image produces a variation in centroid positions causing the ROI to shift back and forth by one pixel. It has been determined that this shift in ROI position can cause multiple discrete centroid clouds if there is sufficient background noise.

To explain this conclusion further, consider the ROI shown in Figure 8.12. Noise on the data causes the ROI to move back and forth between adjacent pixels. This ROI moves by 1 pixel left and right and this is the underlying reason that two groups of centroids can occur. The ROI can be split into three parts based on this movement. Part 2 is the portion of the region that stays constant. Part 1 is the portion of the ROI when it is in the left position and Part 3 is the portion when it is in the right position. The centroids of these three parts can be calculated as are shown in Figure 8.12 by the crosses. When the ROI is in the left position it consists of Part 1 and Part 2 and when it is in the right position it consists of Part 2 and Part 3. In the left position, the centroid is biased to the left by Part 1 and in the right position the centroid is biased to the right by Part 3. Therefore, if the ROI moves due to variation in the centroid position, then this movement can cause the two bunches

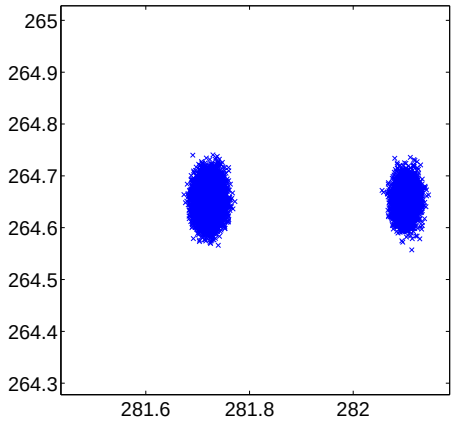


Figure 8.11 Two groups of centroids are shown corresponding to one marker. Two groups can be produced if there is sufficient background noise.

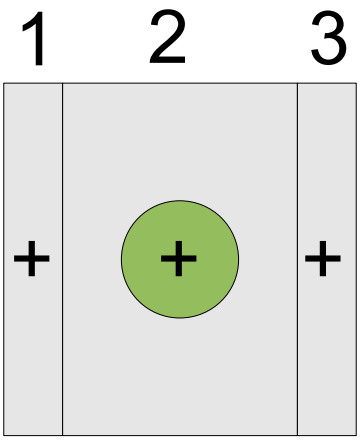


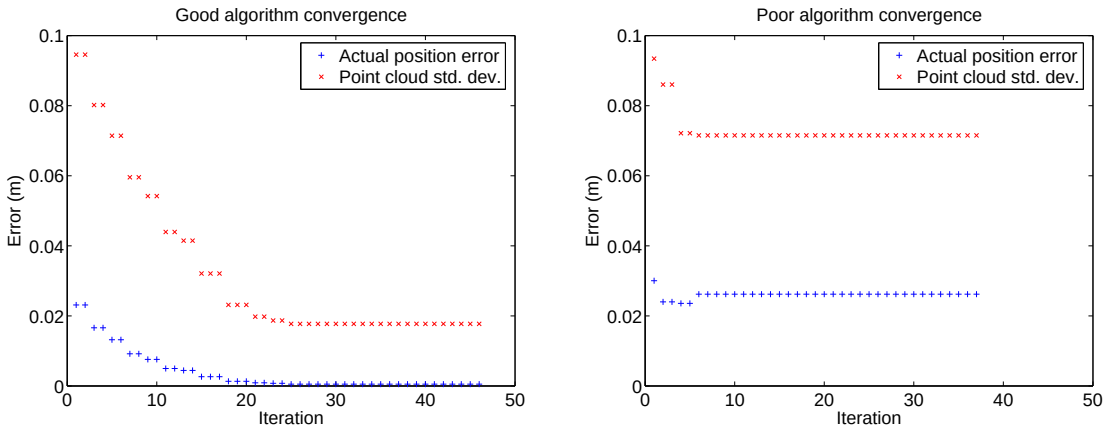
Figure 8.12 A ROI that moves back and forth by one pixel can be broken into three parts. When the ROI is to the left it consists of the left part and the centre part. When the ROI is in the right it consists of the right part and centre part.

of centroids to occur. The image sensor has a black level subtraction mode [78] that allows the average dark pixel value to be subtracted from each pixel. With this enabled, the effects of background noise were reduced substantially [96].

8.3 Pose algorithm testing (3D testing)

This section covers testing the pose algorithm in isolation. In Figure 8.1 this corresponds to testing the system between points 1 and 2. The intention is to determine how well it performs under ideal conditions, using synthetic data. The synthetic data represents the test configuration used in Section 8.1 and does not have any noise overlaid onto it. Figures 8.13a,b show two examples of the the algorithm's behaviour over a number of iterations. Figure 8.13a shows an example of the algorithm converging well. As the number of iterations increases the point cloud variance ($F(q)$, defined in Chapter 7) decreases. As the variance decreases so does the position error. The position error after the final step is approximately 0.005 m. The algorithm has reached a local minimum for $F(q)$ as there is a flat part of the graph where despite the reduction in step size, the algorithm doe not converge further.

Figure 8.13b shows an example of poor performance from the algorithm. In this case the algorithm stops converging early. Although the variance decreases monotonically the position error reaches a minimum and increases slightly again. In this example the minimum position error is 0.026 m. Clearly this is much worse than the first example. Further



(a) (b)

Figure 8.13 The minimisation algorithm's performance varies considerably depending on the initial quaternion used. (a) shows a good example of convergence and (b) shows a bad example. The only difference between these two tests was the initial quaternion used.

research is required to determine the relationship between the initial conditions and the algorithm convergence.

Although the algorithm did not converge in Figure 8.13b, this does not mean that $F(q)$ or the algorithm implementation is incorrect. By setting the initial value of q to the correct value, the variance was determined to be of the order of 10^{-15} m^2. This indicates that there is value of q that will reduce the variance to near zero. Therefore, this value of q could be located if a better performing minimisation routine were implemented.

A graphical representation of the algorithm converging is shown in Figure 8.14. Initially the variance shown by green triangles is large. After each iteration the variance decreases as shown by the decrease in size of the point cloud.

8.4 Summary

These results provide an insight into the stationary performance of the Black Spot modules. Three variables were investigated; namely intensity, distance, and angle. Each of these variables is found to affect the variation in centroid positions. The intensity trend is hyperbolic. An increase in observed marker intensity above approximately 80 at 300 μs has little effect on the variation in centroids, however, below this threshold the standard deviation climbs dramatically. The standard deviation increases approximately linearly with distance but this is a much weaker trend than that of intensity. The angle (position within the FOV) has also been shown to influence the standard deviation. A quadratic curve with a minimum near the centre X coordinate was fitted for the standard deviation versus X coordinate. Data suggests that there is a region between approximately $100 < \mathrm{X} < 500$

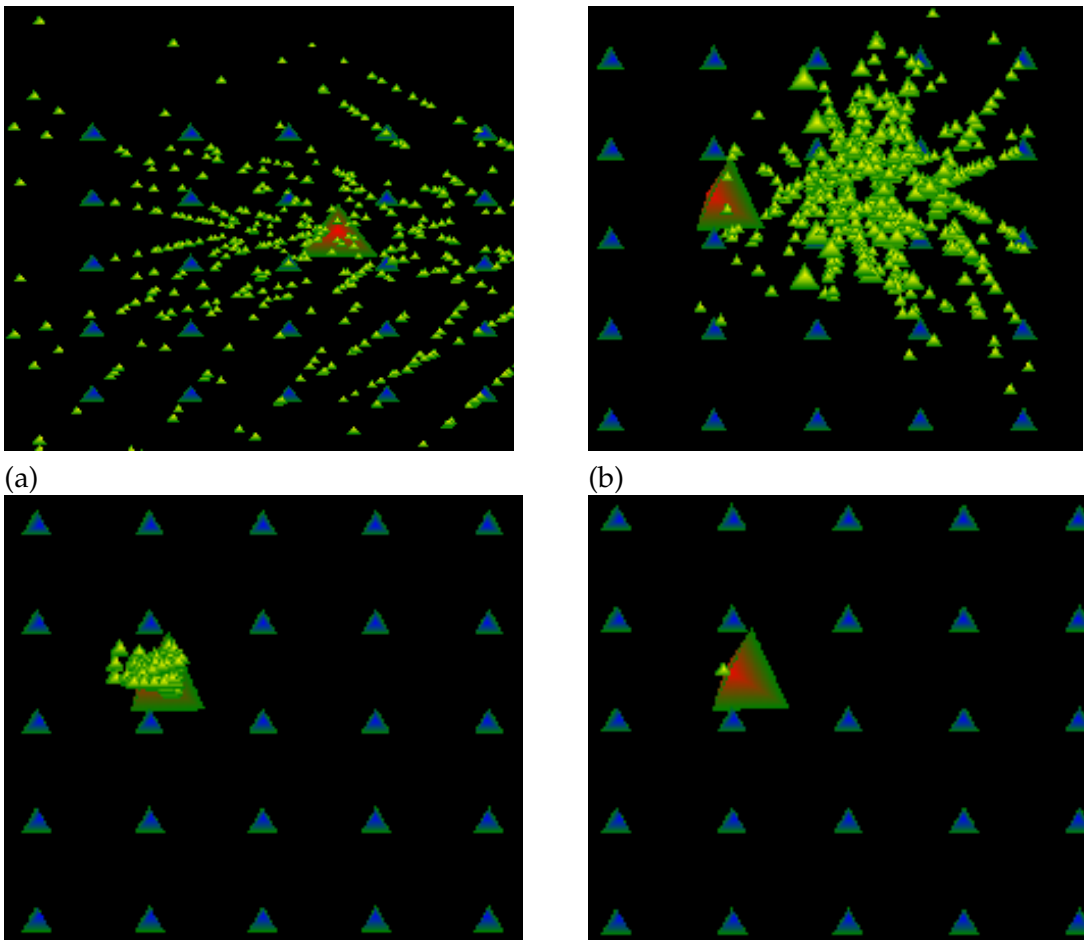


Figure 8.14 The figures show a case where the algorithm converges. The blue-green triangles represent markers. The red-green triangle represents the camera and the green-yellow triangles show the point cloud. a) The initial point cloud. b) The point cloud after the first iteration. c) The same point cloud after 5 iterations. d) The point cloud after 38 iterations.

where the standard deviation is below a 25% increase above the minimum. Outside this region the standard deviations are at least 25% higher than the minimum. The majority of the results obtained are within 50% of the minimum standard deviation regardless of their position in the FOV. The trend curves indicate that at the extremes of the FOV the standard deviations are approximately 0.02 pixels. For comparison, a marker intensity of 10 could result in a standard deviation of approximately 0.1 pixels.

Using the curves fitted to the results in this chapter, a model of the overall standard deviation can be constructed using a combination of each curve. Assuming that the effects are uncorrelated, the model of standard deviation in X can be written

$$\sigma_{\mathrm{X}} = \sqrt{\sigma^2_{\mathrm{X}}(I) + \sigma^2_{\mathrm{X}}(D) + \sigma^2_{\mathrm{X}}(X)}. \tag{8.8}$$

An equation can similarly be written for σ_{Y}.

The performance of the modules were also measured over long periods. The drifts measured indicate that it would be preferable for the system to recalibrate often. Over a period of 2 minutes, the mean standard deviation of a marker near the centre of the FOV was 0.0061 pixels in X and 0.0074 pixels in Y as measured by Module 4. These results correspond to 12.2 $\mu\mathrm{m}$ in X and 14.8 $\mu\mathrm{m}$ in Y at 2 m and agree with Trinder [97] (as reported by Shortis et al. 1995 [48]) who concluded from simulations that subpixel precision of 0.01 pixels or better can be obtained.

The pose algorithm was tested with synthetic data. Under some initial conditions the algorithm converges well. A case is shown where the position error was 0.5 mm. However, under other initial conditions convergence is poorer, at approximately 30 mm. It is concluded that a more robust minimisation algorithm is required to allow the pose to be calculated from a broader range of initial conditions. Lourakis and Argyros [98] note that Levenberg-Marquardt [91] algorithm is used as a minimisation algorithm in structure and motion problems and this suggests it could be of use in the pose estimation algorithm designed.

The Black Spot modules have excellent stationary performance as shown by the results in this chapter and the pose algorithm has potential given a suitable minimisation algorithm. Further testing should be undertaken to reveal the Black Spot modules' dynamic performance. It is apparent that the Blackfin DSP is well suited to marker tracking given the firmware's ability to track 27 markers concurrently. Further testing could investigate how high this number can be increased and incorporate more computational intensive algorithms.

Chapter 9

Conclusions and Future Work

This chapter sums up the outcomes of the project and gives directions for future work.

Overall the project has been successful. A 'Black Spot' camera module was designed and three modules were constructed. An initial firmware implementation was completed showing that each module can track 27 markers at 60 fps. Further firmware development is required. Of particular note is the implementation of the marker quality algorithm. This algorithm could allow the hub to give a lower weighting to markers that have poor centroid standard deviations. Also, due to changes in the internal operation of the firmware, the partitioning algorithm must be re-implemented.

The hardware topology and choice of Blackfin DSP and Aptina CMOS image sensor has proven to work well. However, the initial design required a number of hardware modifications that should be integrated into any new designs. The Blackfin DSP has proven to be fast enough to allow the tracking algorithms to run in real time. However, in the current implementation, the bandwidth of the communications bus is a limiting factor. The number of centroid packets that can be sent is approximately 28 per frame. This limitation is shared between the three camera modules.

The decision not to use SDRAM has allowed functioning modules to be produced that were not a certainty in a design using SDRAM. However, the lack of SDRAM has made firmware development more difficult. Future revisions of the PCB could use SDRAM to allow the firmware to be tested easily.

The PCB's dimensions could be reduced in future revisions by decreasing the sizes of the debug and communications headers and by optimising part placement. Additionally, the Blackfin BF533 DSP is available in a smaller 'chip scale' BGA package. Furthermore, a

future revision of the board must use the CLCC packaged version of the image sensor as this is the only package currently available.

An alternative to the UART bus is the Blackfin SPORT bus that supports hardware TDM. If the SPORT bus were used then a different USB bridge chip would be required to replace the FTDI chip. The Blackfin BF547 [87] could be considered for this job as it includes an on board USB peripheral and SPORT bus. Using this approach, the current bandwidth limit could be increased. Alternatively, the throughput could be increased by sending compressed binary data.

The Soft Hub processes the data from the Black Spot modules. A rudimentary pose estimate for single modules under some conditions can be produced. However, the minimisation algorithm does not converge sufficiently under many conditions. Future work should investigate replacing this algorithm with a bundle adjustment algorithm [51]. This would allow the model of the markers' 3D locations to be refined while producing pose estimates. Lourakis and Argyros describe the design of a bundle adjustment package [98] and provide software licensed under the GNU General Public Licence (GPL) [99].

The stationary centroid performance of the Black Spot modules is excellent. The modules were tested while varying three parameters: angle, distance, and marker intensity. The variation in marker intensity had the largest impact on centroid standard deviation but with a shutter period of 300 μs, and maximum intensity values of greater than 80, similar results are produced. With camera-to-beacon distances of between 0.6 m and 3 m there is a slight trend towards greater variation with increasing distance. The centroid standard deviation varies within a module's FOV with the best performance near the centre of the FOV. Over a period of 2 minutes, the mean standard deviations of a marker near the centre of the FOV were 0.0061 pixels in X and 0.0074 pixels in Y. It is concluded that the centroid calculation exhibits excellent precision.

The implementation of the firmware could be improved. For example, the image capture driver produces erroneous data when the firmware is run from the flash memory. To overcome this, the firmware must be loaded using the debugger. Another issue occurs when the hub uses the flow control line to instruct a Black Spot module to halt transmission. This can cause a buffer overrun in the image sensor driver while the Black Spot is waiting for the hub. The buffer overrun could be avoided by using a Real Time Operating System (RTOS) to allow the image processing routines to run in a different thread from the communications routines. Another option is to service the driver using a callback from an ISR that is driven by a timer peripheral. The first method is preferable over the second as it is a more general solution and would make extending the firmware easier.

The four major improvements required to continue this project are to improve the robust-

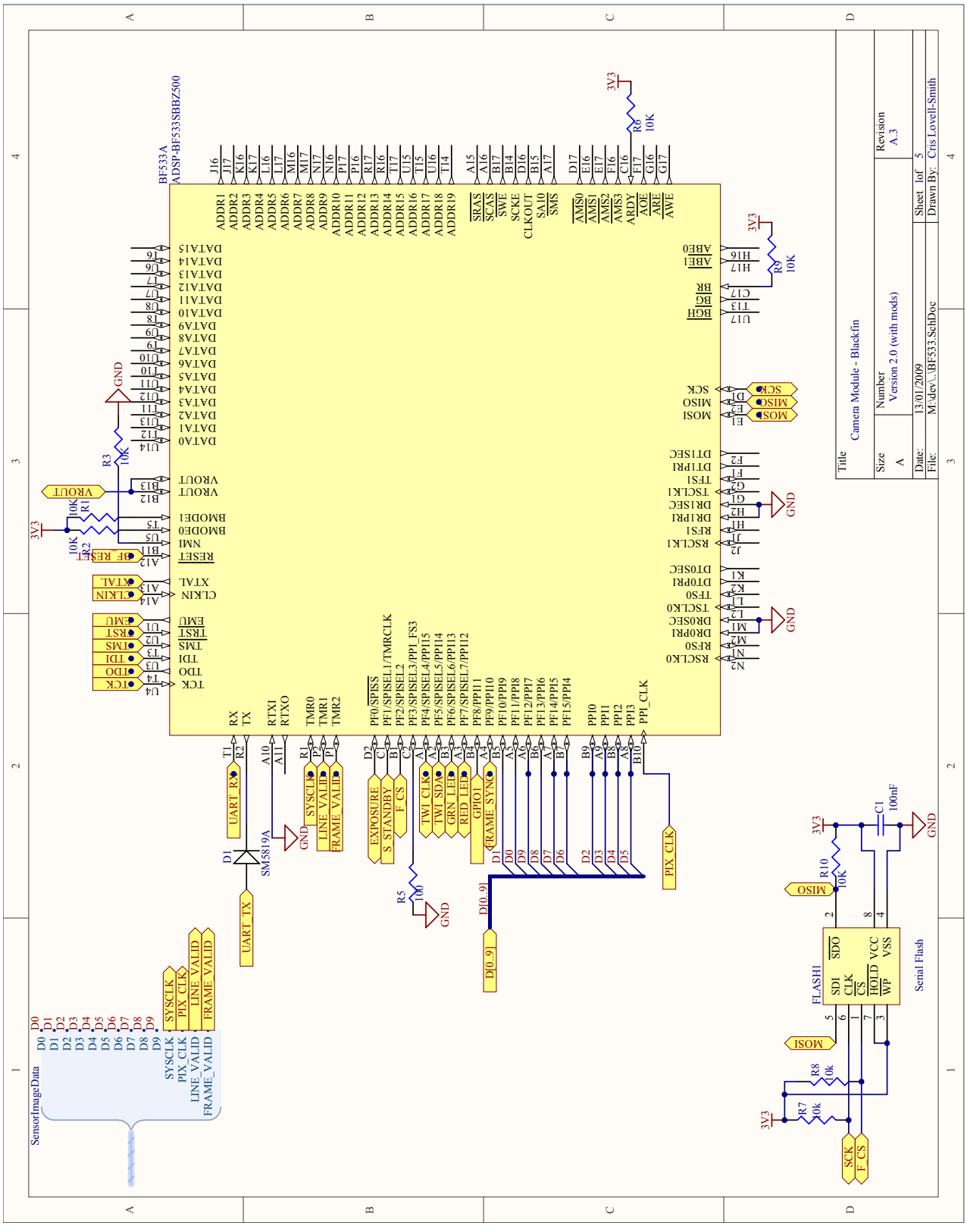
ness of the firmware, increase the communications bus bandwidth, implement a bundle adjustment algorithm, and to calibrate the system. The first two steps are both implementation details and are achievable through firmware and hardware changes. The last two steps may require significant further research. Initially, the bundle adjustment software provided by Lourakis and Argyros could be investigated to determine how suitable their approach is. System calibration could be partially solved by the bundle adjustment method. However, offline lens calibration may prove advantageous and the method proposed by Zhang [28,52] could be used. Finally, calibration must also include the estimation of the camera modules' poses within the hub enclosure, and possibly the construction of marker models. Marker models should describe how a centroid is biased with respect to the angle it is viewed upon.

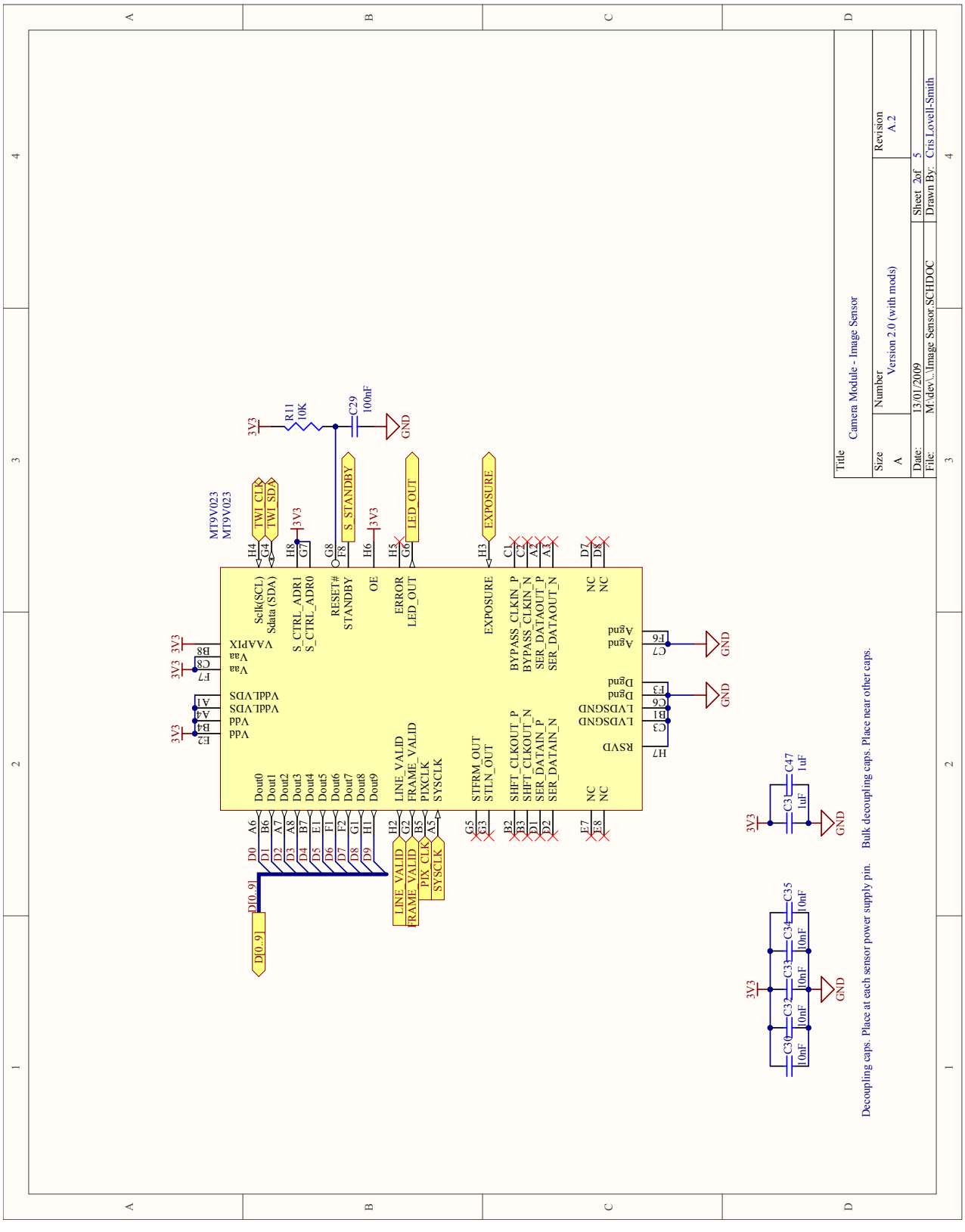
In summary, the Black Spot modules have potential to become part of a future optical tracking system with further research and development.

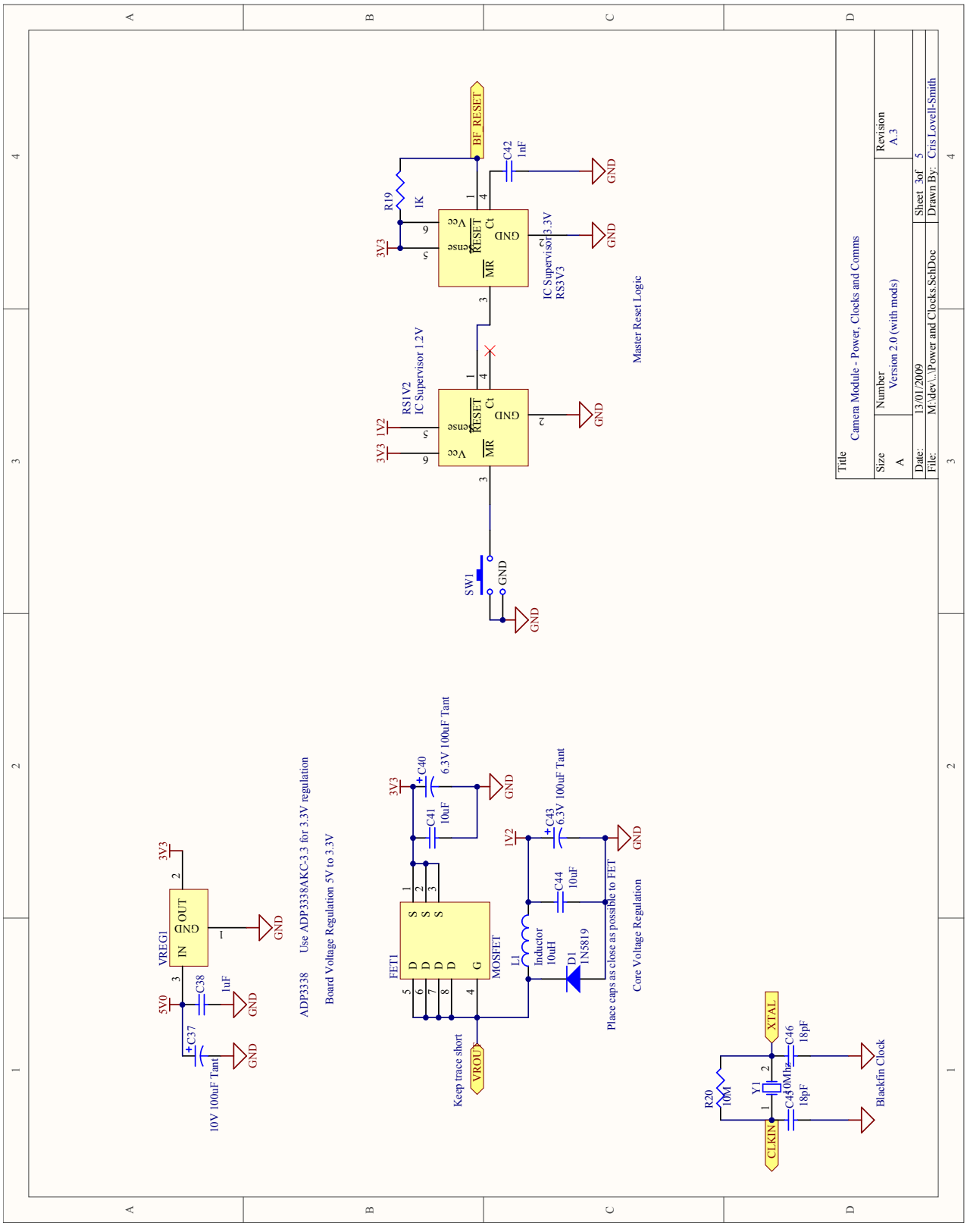
Appendix A

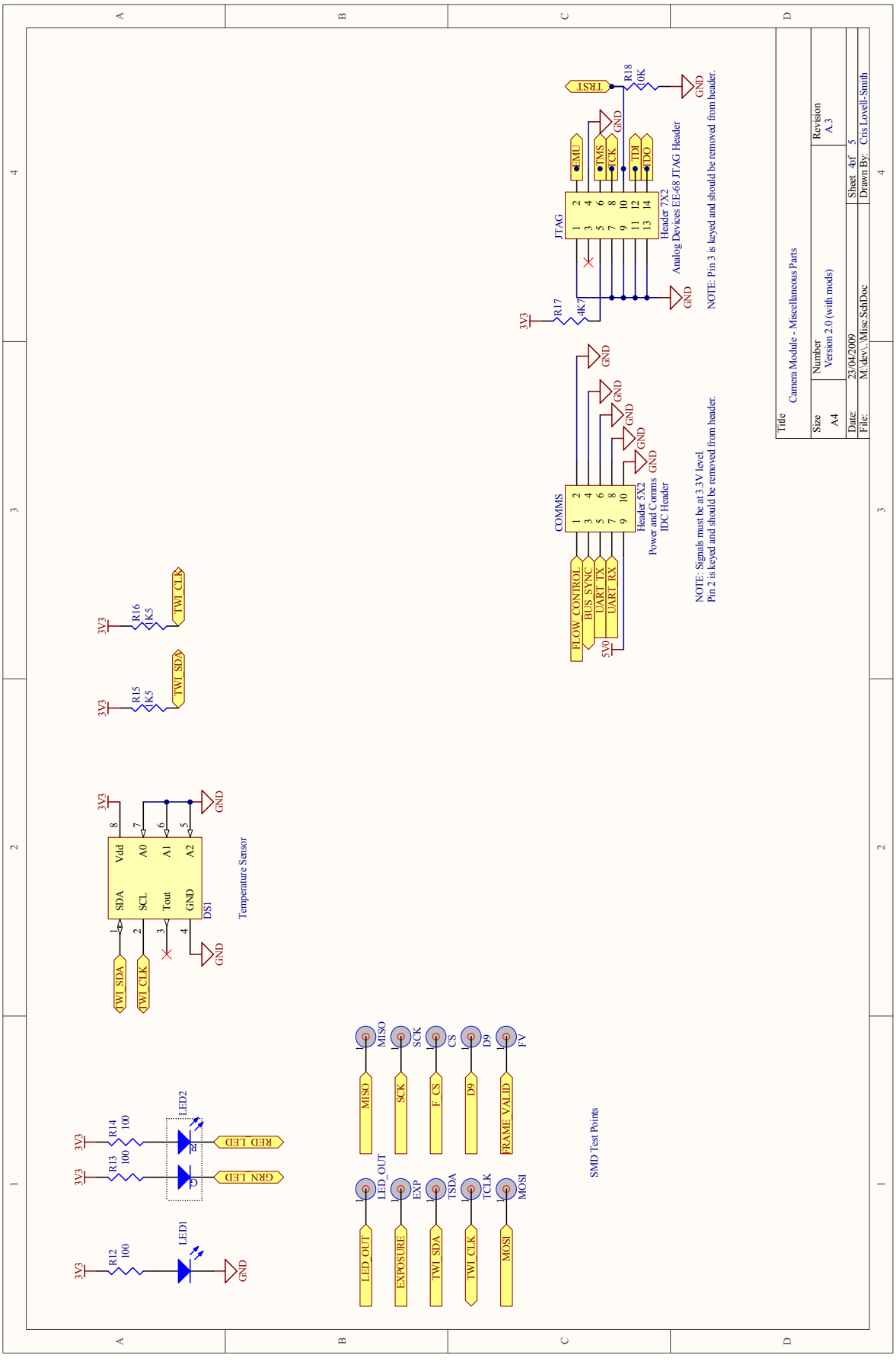
Black Spot Schematics

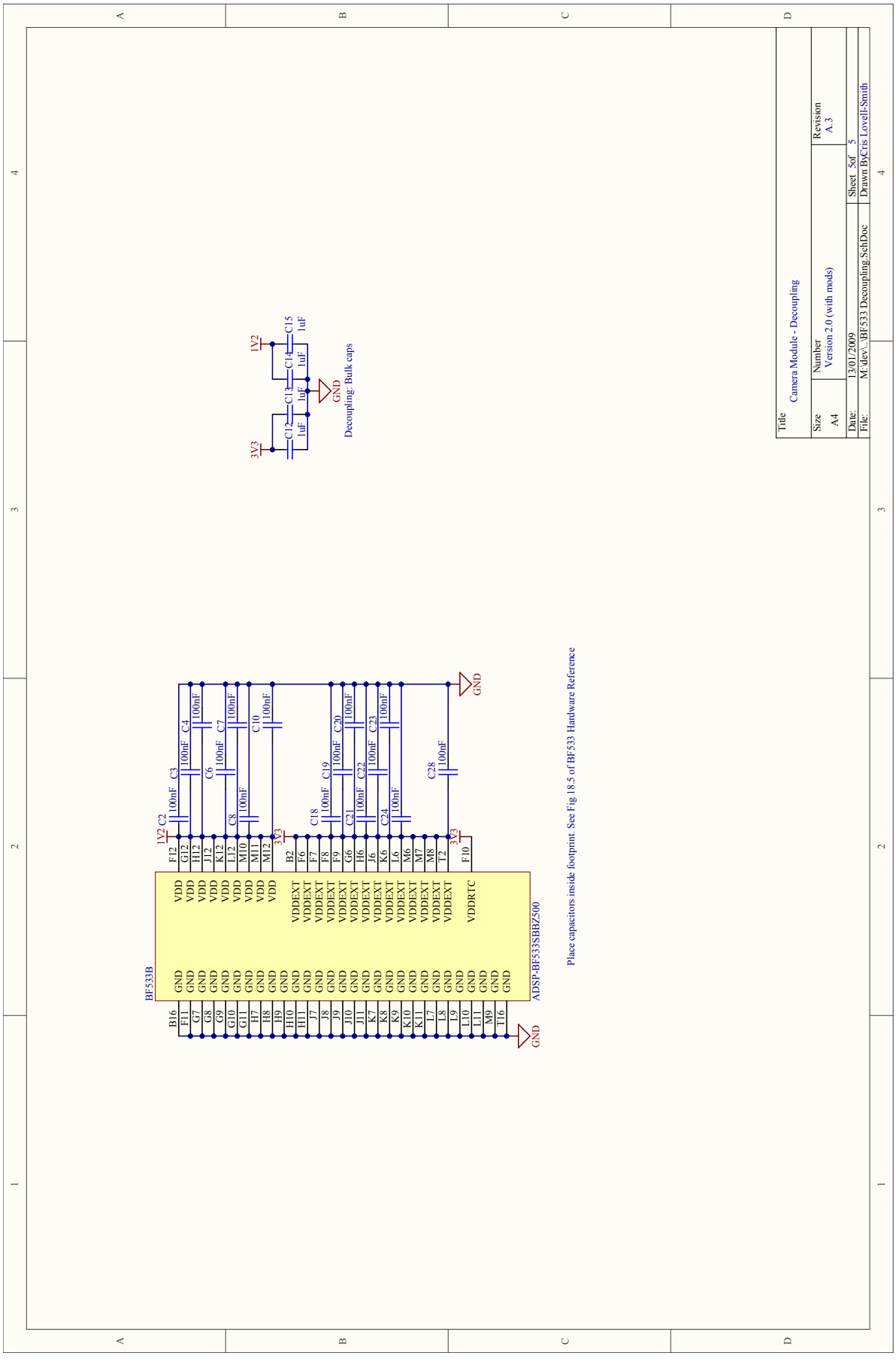
The Black Spot schematic is reproduced in this appendix. The schematic is shown in its modified form that fixes the initial errors in the circuit design.


BF533A
ADSP-BF533SBBZ500
Camera Module - Blackfin
Number
Version 2.0 (with mods)
Revision
A.3
13/01/2009
Sheet 1of 5
M:\dev\...\BF533.SchDoc
Drawn By: Cris Lovell-Smith
SensorImageData
Serial Flash
FLASH1
UART_RX
UART_TX
SM5819A
SYSCLK
LINE_VALID
FRAME_VALID
PIX_CLK
EXPOSURE
S_STANDBY
F_CS
TWI_CLK
TWI_SDA
GRN_LED
RED_LED
GPIO1
FRAME_SYNC
D[0..9]
MOSI
MISO
SCK
3V3
GND
VROUT
BF_RESET


MT9V023
MT9V023
Scl/k(SCL)
Sdata (SDA)
S_CTRL_ADR1
S_CTRL_ADR0
RESET#
STANDBY
OE
ERROR
LED_OUT
EXPOSURE
BYPASS_CLKIN_P
BYPASS_CLKIN_N
SER_DATAOUT_P
SER_DATAOUT_N
NC
NC
Agnd
Dgnd
LVDSGND
RSVD
VAAPIX
Vaa
VddL_VDS
Vdd
Dout0
Dout1
Dout2
Dout3
Dout4
Dout5
Dout6
Dout7
Dout8
Dout9
LINE_VALID
FRAME_VALID
PIXCLK
SYSCLK
STFRM_OUT
STLN_OUT
SHFT_CLKOUT_P
SHFT_CLKOUT_N
SER_DATAIN_P
SER_DATAIN_N
TWI_CLK
TWI_SDA
S_STANDBY
LED_OUT
EXPOSURE
R11
10K
C29
100nF
3V3
GND
D[0..9]
LINE_VALID
FRAME_VALID
PIX_CLK
SYSCLK
C47
1uF
C30
C32
C33
C34
C35
10nF
Decoupling caps. Place at each sensor power supply pin.
Bulk decoupling caps. Place near other caps.
Title
Camera Module - Image Sensor
Size
A
Number
Version 2.0 (with mods)
Revision
A.2
Date:
13/01/2009
Sheet 2of 5
File:
M:\dev\..\Image Sensor.SCHDOC
Drawn By:
Cris Lovell-Smith


Title
Camera Module - Power, Clocks and Comms
Size
A
Number
Version 2.0 (with mods)
Revision
A.3
Date:
13/01/2009
Sheet 3of 5
File:
M:\dev\...\Power and Clocks.SchDoc
Drawn By:
Cris Lovell-Smith
Master Reset Logic
BF RESET
R19
1K
C42
1nF
IC Supervisor 3.3V
RS3V3
3V3
GND
RS1V2
IC Supervisor 1.2V
1V2
SW1
Core Voltage Regulation
FET1
MOSFET
C40
6.3V 100uF Tant
C41
10uF
L1
Inductor
10uH
C43
6.3V 100uF Tant
C44
10uF
D1
1N5819
Place caps as close as possible to FET
Keep trace short
VROUT
VREG1
C37
10V 100uF Tant
C38
1uF
5V0
ADP3338
Use ADP3338AKC-3.3 for 3.3V regulation
Board Voltage Regulation 5V to 3.3V
R20
10M
Y1
C45
18pF
C46
18pF
XTAL
CLKIN
Blackfin Clock


Title
Camera Module - Miscellaneous Parts
Size
A4
Number
Version 2.0 (with mods)
Revision
A.3
Date:
23/04/2009
Sheet 4of 5
File:
M:\dev\..\Misc.SchDoc
Drawn By:
Cris Lovell-Smith
SMD Test Points
LED_OUT
EXPOSURE
TWI_SDA
TWI_CLK
MOSI
MISO
SCK
F_CS
D9
FRAME_VALID
LED1
LED2
R12 100
R13 100
R14 100
RED_LED
GRN_LED
Temperature Sensor
DS1
TWI_SDA
TWI_CLK
R15 K5
R16 K5
COMMS
FLOW_CONTROL
BUS_SYNC
UART_TX
UART_RX
5V0
Header 5X2
Power and Comms IDC Header
NOTE: Signals must be at 3.3V level.
Pin 2 is keyed and should be removed from header.
JTAG
R17 4K7
R18 0K
TRST
EMU
TMS
TCK
TDI
TDO
Header 7X2
Analog Devices EE-68 JTAG Header
NOTE: Pin 3 is keyed and should be removed from header.


Title
Camera Module - Decoupling
Size
A4
Number
Version 2.0 (with mods)
Revision
A.3
Date:
13/01/2009
Sheet 5of 5
File:
M:\dev\..\BF533 Decoupling.SchDoc
Drawn By:
Cris Lovell-Smith
Decoupling: Bulk caps
Place capacitors inside footprint. See Fig 18.5 of BF533 Hardware Reference
ADSP-BF533SBBZ500
BF533B
1V2
3V3
GND
VDD
VDDEXT
VDDRTC

Appendix B

Black Spot PCB

The top side of the Black Spot circuit board is shown in Figure B.1 followed by the bottom side of the board in Figure B.2.

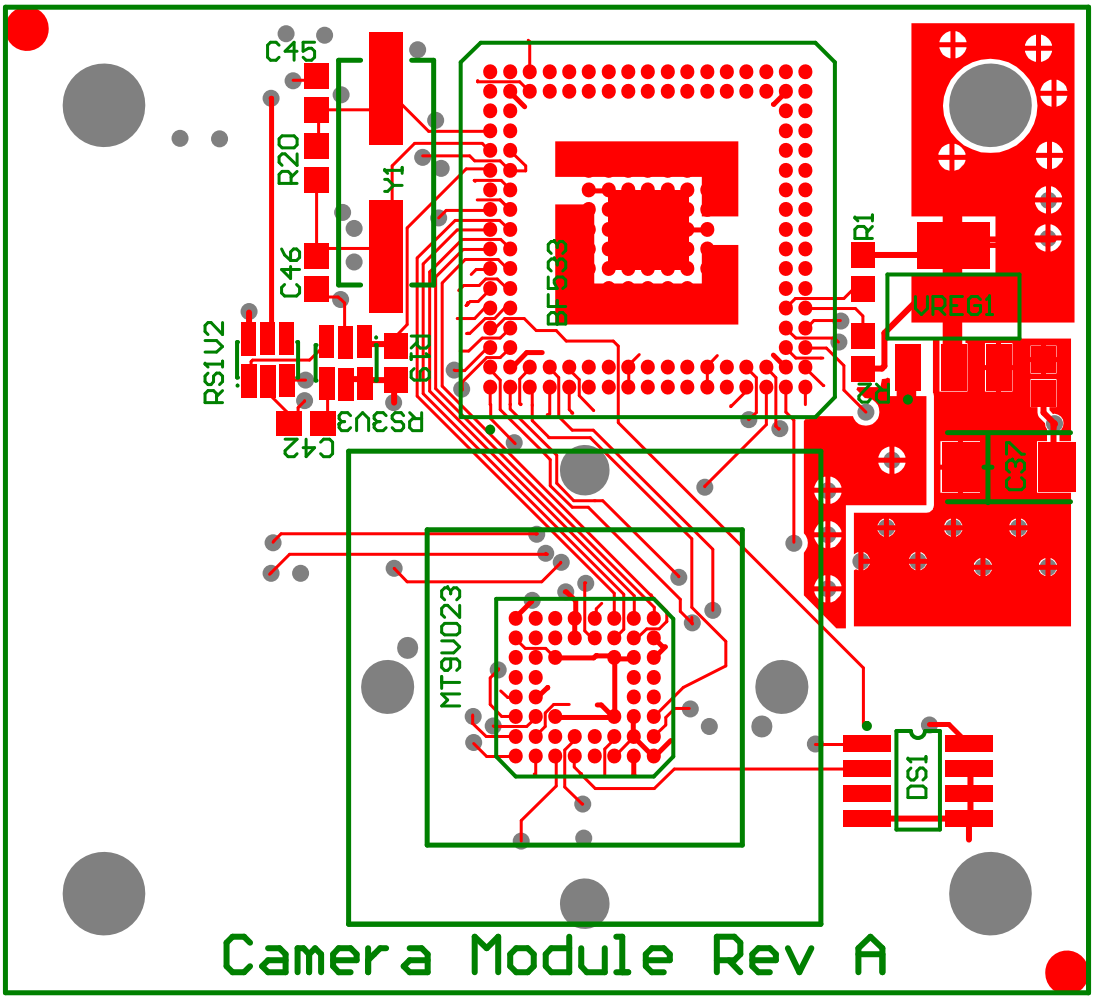


Figure B.1 The top side of the Black Spot PCB showing the top copper layer (red), silk screen (green), and drill holes and vias (gray).

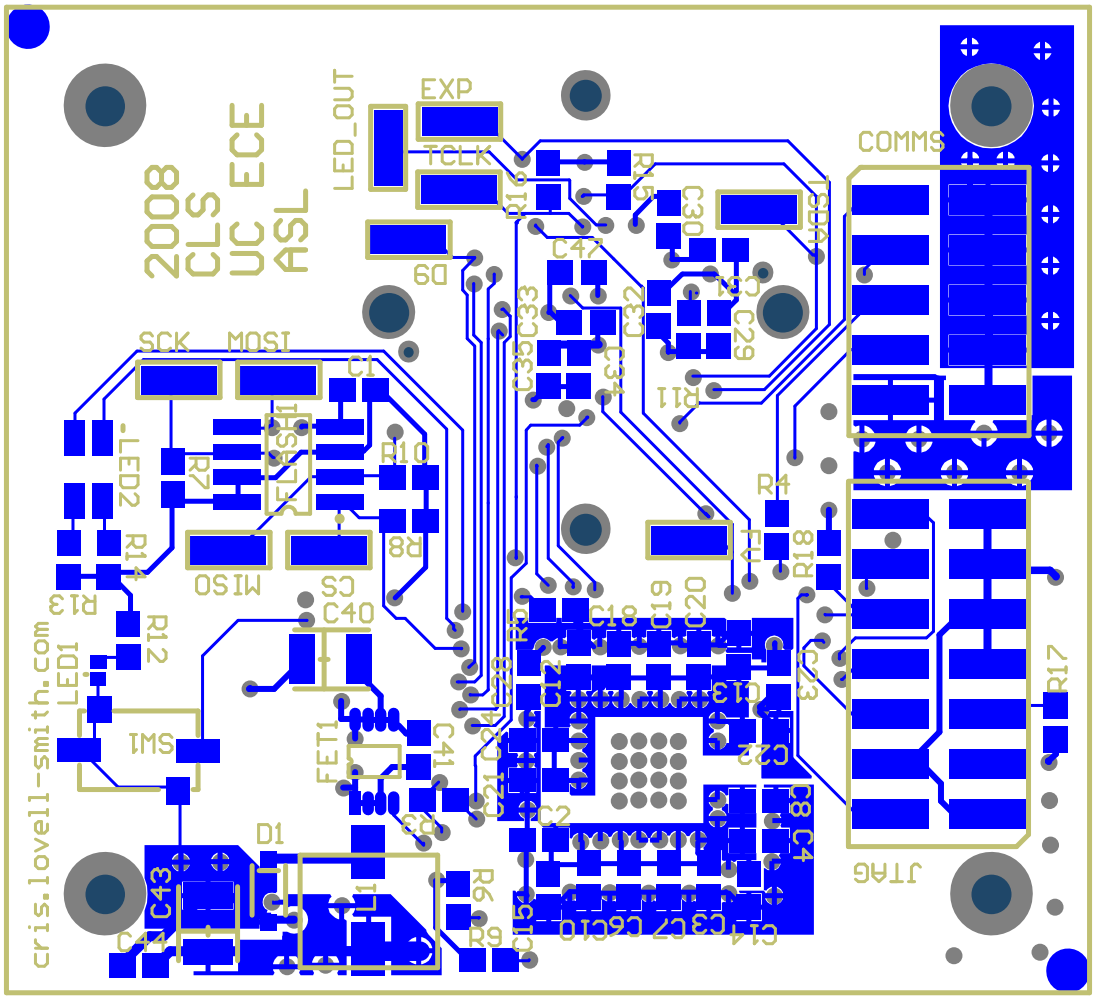
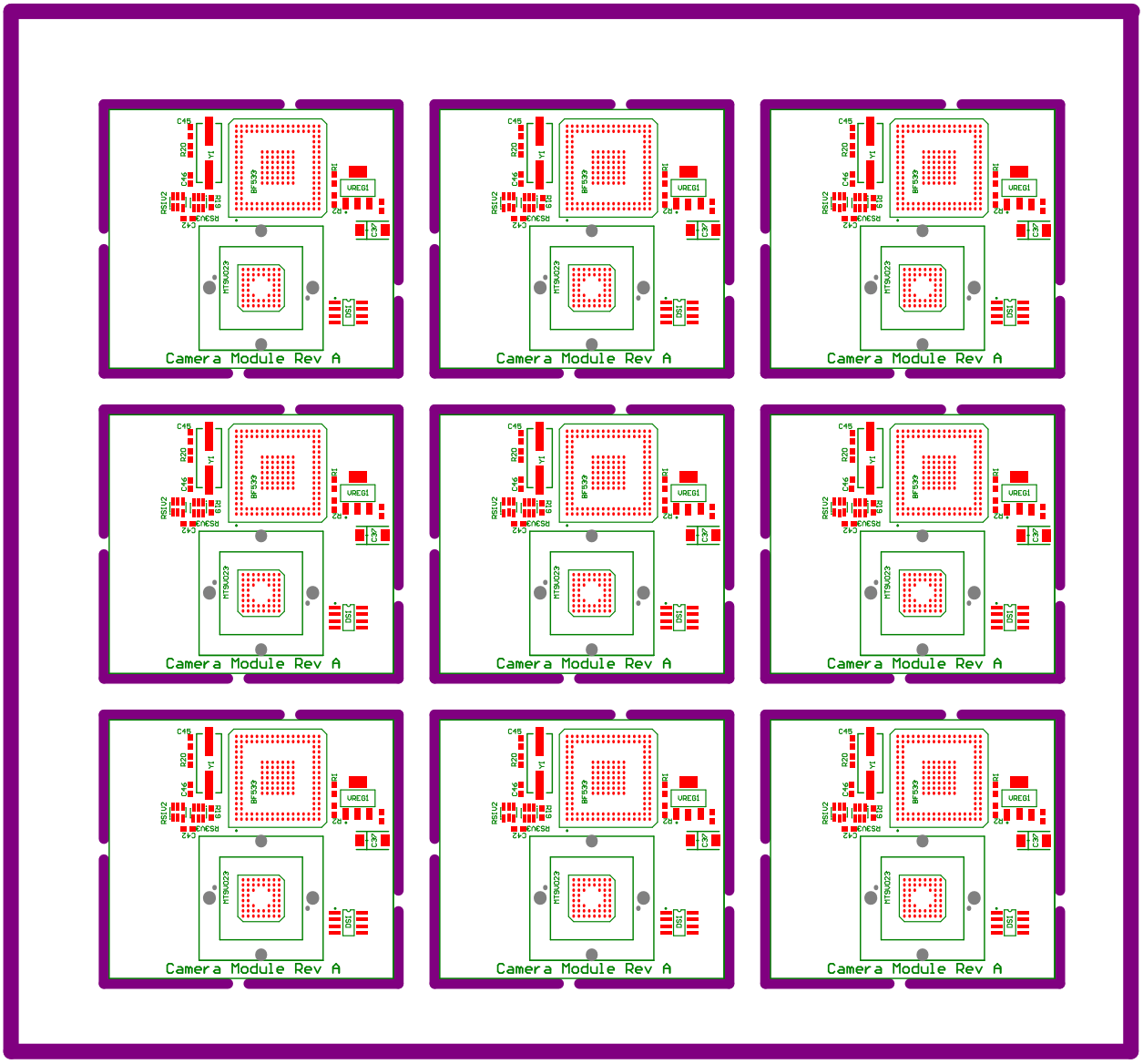


Figure B.2 The bottom side of the Black Spot PCB showing the top copper layer (red), silk screen (green), and drill holes and vias (gray).

Appendix C

Black Spot PCB Panel

An array of Black Spot PCBs placed in a panel as was submitted to the manufacturer is shown below.


Camera Module Rev A

Appendix D

Black Spot PCB Modifications

D.1 Buck converter modification

The initial design had a short between the 5V supply and ground. To alter the PCB to match the correct schematic, lift pins 5-8 of FET1. Connect a wire between the cathode of D1 to all lifted pins of FET1 as shown in Figure D.1.

D.2 Flash select line

The flash select line must be connected to processor flag PF2. This was initially connected to another line. To fix this cut the track running to pin 1 of FLASH1. Rewire pin 1 of FLASH1 to the via to the left of the R11 designator as shown in Figure D.2.

D.3 Image data bus modification

The image data bus must be rewired to allow 8 bit reads of the image sensor to be initiated without the possibility of the bright parts of the image producing a banding effect caused by the least significant bits resetting to zero. This operation is difficult and involves rewiring each image sensor data line so that the effect is a logic roll right by 2 bits. The schematic given in Appendix A is the revised schematic and shows the modified image data bus. To modify the PCB to match the schematic requires cutting tracks on the top and bottom of the board.

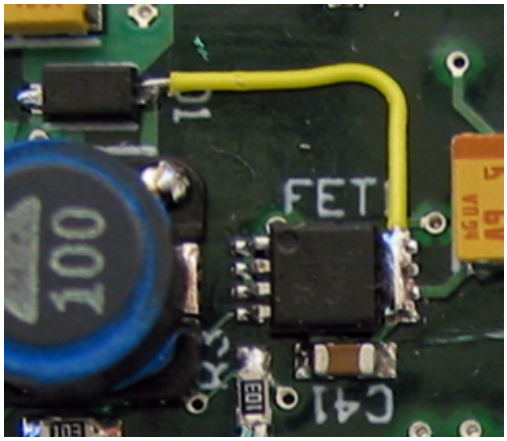


Figure D.1 To remove the short to ground on the buck converter, lift pins 5-8 of FET1 and wire to the cathode of D1.

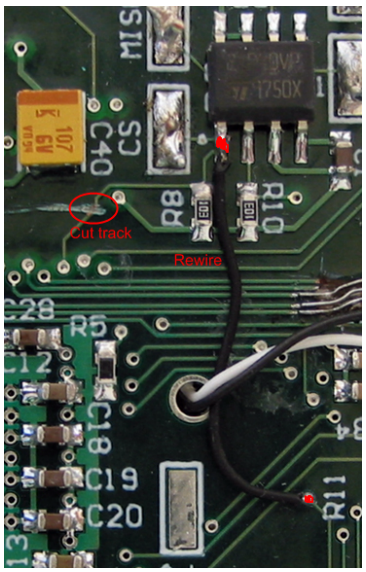


Figure D.2 To enable the flash select line, cut the track shown and rewire between pin 1 of FLASH1 and the via to the left of R11.

D.4 Image sensor reset

This modification fixes a defective RC filter in the initial design. The board seems to work fine without this modification. Figure D.3 shows the modification. Cut the track as shown and rewire.

D.5 UART transmit modification

Unfortunately the Blackfin BF533 cannot put the UART transmit line into a high impedance state. This makes sharing the communications bus with other modules more difficult. To overcome this problem the current limiting resistor in series with Black Spot module's transmit line can be replaced with a Shottky diode. Also, the communications bus requires one pullup resistor (10K) between transmit and the 3.3 V supply. Figure D.4 shows the Black Spot board. The Shottky diode replaces resistor R4 and its orientation is shown. The circuit diagram for the pullup resistor is shown in Figure 5.8.

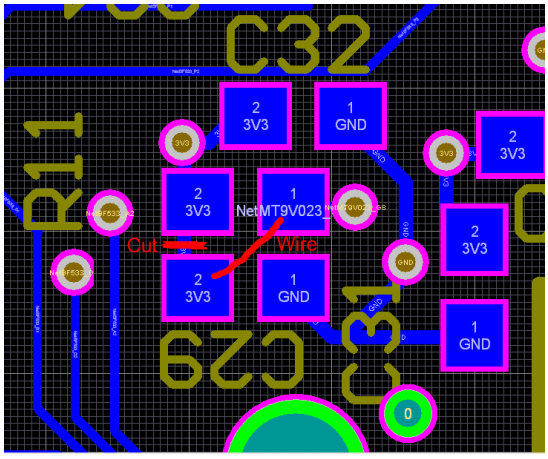


Figure D.3 A defective RC filter can be rewired as shown.

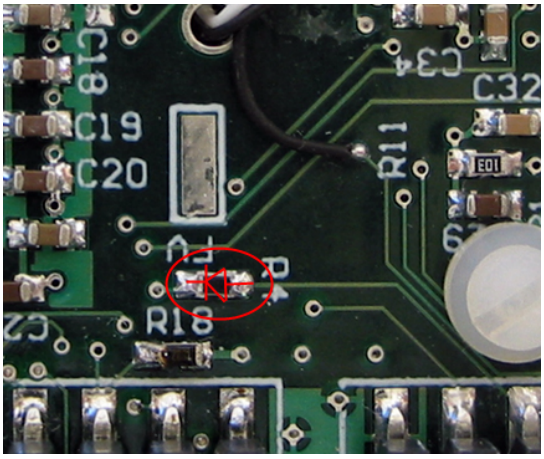


Figure D.4 The resistor R4 must be replaced with a Shottky diode to allow the Black Spot modules' to share the bus.

Appendix E

Black Spot Bill of Materials

Designator	Name	Description	Digikey Part #	Value	Quantity
BF533	Blackfin DSP	Blackfin Embedded Processor, 169-Pin PBGA, Lead-Free, 500 MHz, Industrial	ADSP-BF533SBBZ500-ND		1
C42	Capacitor	Capacitor (Semiconductor SIM Model)		1nF	1
C12, C13, C14, C15, C31, C38, C47	Capacitor	Capacitor (Semiconductor SIM Model)		1uF	7
C30, C32, C33, C34, C35	Capacitor	Capacitor (Semiconductor SIM Model)		10nF	5
C41, C44	Capacitor	Capacitor (Semiconductor SIM Model)		10uF	2
C45, C46	Capacitor	Capacitor (Semiconductor SIM Model)		18pF	2
C1, C2, C3, C4, C6, C7, C8, C10, C18, C19, C20, C21, C22, C23, C24, C28, C29	Capacitor	Capacitor (Semiconductor SIM Model)		100nF	17
C37	Tantulum Capacitor	Tantulum Capacitor Case Size C	495-2221-1-ND	10V 100uF Tant	1
C40, C43	Tantulum Capacitor	Tantulum Capacitor Case Size B	495-2191-1-ND	6.3V 100uF Tant	2
COMMS	Header 5X2	Header, 5-Pin, Dual row			1
D1	1N5819 Diode	1A Surface Mount Shottky Barrier Rectifier	1N5819HW-FDICT-ND		1
DS1	TWI Thermometer	High-Precision Digital Thermometer and Thermostat	DS1631Z+-ND		1
FET1	MOSFET	Enhancement MOSFET	ZXM64P02XCT-ND		1
FLASH1	SPI Flash	4 MBit, Low voltage, serial SPI flash 50Mhz	497-3598-ND		1

Designator	Name	Description	Digikey Part #	Value	Quantity
JTAG	Header 7X2	Header, 7-Pin, Dual row	TSM-110-01-T-DV-ND		1
L1	Inductor	Inductor	445-2014-1-ND	10uH	1
LED1	LED SMT Blue	Typical BLUE SiC LED	511-1615-1-ND		1
LED2	SMT LED Red-Green	SMT LED Red-Green	754-1076-1-ND		1
MT9V023	MT9V023	Aptina CMOS Image Sensor	557-1401-ND		1
R19	Resistor			1K	1
R15, R16	Resistor			1K5	2
R17	Resistor			4K7	1
R1, R2, R6, R7, R8, R9, R10, R11, R18	Resistor			10K	9
R20	Resistor			10M	1
R5, R12, R13, R14	Resistor			100	4
RS1V2	IC Supervisor 1.2V	IC Supervisor 1.2V	296-17190-1-ND		1
RS3V3	IC Supervisor 3.3V	IC Supervisor 3.3V	296-17195-1-ND		1

Table E.1 The Black Spot Bill of Materials.

Appendix F

Communications Packets

Common packets used by the Black Spot modules and Soft Hub for communications are listed in this appendix.

F.1 Centroid Packet (C)

Sent by a camera module to the hub to update a known marker's centroid.

Fields:

Field Name	Field Description	Field Type	Field Size in Bits
Destination ID	The Soft Hub's ID.	Integer	8
Timestamp	An integer representing the time that the packet was constructed.	Integer	32
Sender ID	The module's ID.	Integer	8
Packet ID	A 'C' for centroid.	Character	8
Centroid ID	A unique ID for the centroid.	Integer	16
ROI X	The X co-ordinate of the upper left corner of the ROI (pixels).	Integer	16
ROI Y	The Y co-ordinate of the upper left corner of the ROI (pixels).	Integer	16
X Numerator	The numerator in the fraction that describes the X coordinate, i.e., $X = \frac{\text{X Numerator}}{\text{Pixel Sum}}$.	Integer	32
Y Numerator	The numerator in the fraction that describes the Y coordinate, i.e., $Y = \frac{\text{Y Numerator}}{\text{Pixel Sum}}$.	Integer	32
Pixel Sum	The sum of pixel values that are above the signal threshold within the ROI.	Integer	32
Pixel Count	The number of pixels that are above the signal threshold in the ROI.	Integer	16
ROI Width	The ROI width (pixels).	Integer	8
ROI Height	The ROI height (pixels).	Integer	8
Max	The maximum pixel intensity of the marker image.	Integer	8

F.2 Information Packet (INFO)

Sent by modules to the hub to transfer a) debug information b) text to the user.

Fields:

Field Name	Field Description	Field Type	Field Size in Bits
Data	The information in a text format.	String	N/A

F.3 Marker Lost (ML)

Sent by a module to the hub when a marker is lost. This occurs when the camera module movement is too great causing the marker image to no longer be within the bounds of its ROI. More commonly this occurs when the marker image leaves the camera module's FOV.

Fields:

Field Name	Field Description	Field Type	Field Size in Bits
Centroid ID	The ID of the centroid within the ROI that was lost.	Integer	16

F.4 Request Data Packet (RQ)

Sent by the hub to a module to indicate that it should use the bus to transfer any data it has waiting for the hub.

Fields: Header only.

F.5 Request All Data Packet (RQA)

Sent by the hub to the broadcast address (all modules) to indicate that they should begin sending their data in the time slots set using the STDM packet

Fields: Header only.

F.6 Set Time Slot Packet (STDM)

Sent by the hub to a module to indicate the time slot that the module receiving the packet should use.

Fields:

Field Name	Field Description	Field Type	Field Size in Bits
Slot Number	The slot number.	Integer	8
Number of Slots	The number of time slots. This is equal to the number of camera modules attached to the bus.	Integer	8

F.7 Data Transfer complete (TXC)

Sent by a camera module to indicate that it has sent all of the data it had to the hub. This is sent just before the module releases the bus for the next module to use.

Fields: Header only.

Appendix G

Outlier Analysis

Spikes were found in the long term data trend. It is assumed that these spikes correspond to the movement near the test site. For example, there is a railway line that passes nearby and throughout the day trains use the line. To test this theory a measurement was made while a train was passing. The measurement started while the train was near the office and finished after it could no longer be heard. The entire measurement took less than two minutes so two minute intervals should not be used to analyse the data as was done in Section 8.2.5. Instead intervals of 100 samples were used corresponding to approximately 1.5 seconds. Figure G.1 shows the standard deviation in Y of one of the marker's centroids. As expected, the standard deviation is high while the train is passing and drops away as the train leaves. This was expected as vibrations can be felt at the test site while trains are moving nearby.

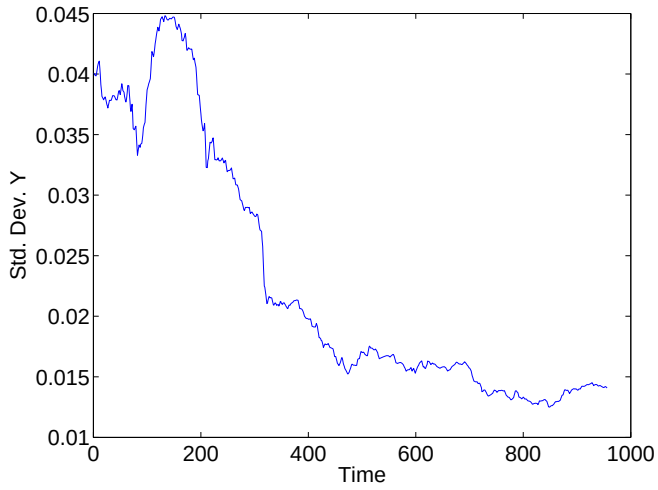


Figure G.1 A train passing the testing site causes a spike in standard deviation. Here the effects of the train can be seen passing at the beginning of the measurement. This causes spikes in the standard deviation of a window of 100 samples. As the train passes the standard deviation in the Y coordinates drops. The shape of this graph is important as it shows high values while the train is nearby and low values after the train has passed.

References

[1] (2009, April) Polhemus fastscan used in the lord of the rings! Polhemus. [Online]. Available: http://www.polhemus.com/polhemus_editor/assets/LOTR.pdf

[2] Polhemus. (2009, April) Hanger has revolutionized the process of fitting a patient for a prosthetic or orthotic device with insignia. WWW. Polhemus. [Online]. Available: http://www.polhemus.com/polhemus_editor/assets/Hanger%20_FastSCAN.pdf

[3] (2009, April) Compumedics neuroscan case study: Polhemus fastrak enables neuroimaging for medical research. WWW. Polhemus. [Online]. Available: http://www.polhemus.com/polhemus_editor/assets/NeuroScan%20_Fastrak_.pdf

[4] (2009, April) Gentle giant studios chooses the polhemus fastscan. Polhemus. [Online]. Available: http://www.polhemus.com/polhemus_editor/assets/GentleGiant.pdf

[5] (2009, April) Fastscan lets you race through reverse engineering. Polhemus. [Online]. Available: http://www.polhemus.com/polhemus_editor/assets/MWDesign%20Corvette%20conversion.pdf

[6] (2009, April) Fastscan was the scanner of choice for this restoration woodworking project. [Online]. Available: http://www.polhemus.com/polhemus_editor/assets/honeoye-falls-millwork2.pdf

[7] R. A. Johnston, K. Barnes, T. Lovell-Smith, and N. B. Price, "Use of a hand-held laser scanner in palaeontology: A 3d model of a plesiosaur fossil," *Image and Vision Computing New Zealand*, 2004.

[8] (2009, April) Museum conservation institute joint mongolian-smithsonian deer stone project. WWW. Smithsonian - Museum Conservation Institute. [Online]. Available: http://www.si.edu/mci/english/research/conservation/deer_stones.html

[9] Polhemus, *Fastrak - 3space Fastrak User's Manual*, Polhemus, June 2004.

[10] *FastSCAN Cobra and Scorpion - Handheld Laser Scanner User Manual*, 1st ed., ARANZ Scanning LTD., Unit 4, 15 Washington Way, Sydenham, Christchurch 8011, New Zealand, January 2009.

[11] M. A. Nixon, B. C. McCallum, W. R. Fright, and N. B. Price, "The effects of metals and interfering fields on electromagnetic trackers," *Presence: Teleoper. Virtual Environ.*, vol. 7, no. 2, pp. 204–218, 1998.

[12] G. Welch and E. Foxlin, "Motion tracking: No silver bullet, but a respectable arsenal," *IEEE Computer Graphics and Applications*, vol. 22, no. 6, pp. 24–38, 2002.

[13] (2009, April) Faro international website. Faro International. [Online]. Available: www.faro.com

[14] (2009, April) Intersense - technical overview is-900 motion tracking system. Web. Intersense. [Online]. Available: http://www.isense.com/uploadedFiles/Products/White_Papers/IS900_Tech_Overview_Enhanced.pdf

[15] (2009, February) Global positioning system - serving the world. http://www.gps.gov. [Online]. Available: http://www.gps.gov

[16] G. Welch, G. Bishop, L. Vicci, S. Brumback, K. Keller, and D. Colucci, "High-performance wide-area optical tracking: The hiball tracking system," 2001.

[17] (2009, April) Advanced realtime tracking gmbh website. Advance Realtime Tracking GmbH. [Online]. Available: http://www.ar-tracking.de/

[18] (2009, April) Industrial research limited website. [Online]. Available: www.irl.cri.nz

[19] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME Journal of Basic Engineering*, no. 82 (Series D), pp. 35–45, 1960.

[20] (2008, Nov.) A.r. tracker flyer. [Online]. Available: http://www.ar-tracking.de/fileadmin/redakteure/Flyer_und_Logos/Flyer.pdf

[21] R. J. Valkenburg, J. Penman, J. Schoonees, N. Alwesh, and G. Palmer, "Interactive hand-held 3d scanning," in *International Vision Computing New Zealand (IVCNZ)*, 2006.

[22] R. Valkenburg, N. Alwesh, Y. Zhao, and R. Klette, "Iterative target calibration using conformal geometric algebra," in *International Vision Computing New Zealand (IVCNZ)*, 2006.

[23] 3rd tech website. [Online]. Available: www.3rdtech.com

[24] (2009, April) Ascension laserbird 2. Web. Fifth Dimension Technologies. [Online]. Available: http://www.5dt.com/products/plaserbird.html

[25] (2009, April) Optotrak proseries - portable metrology system. [Online]. Available: http://www.ndigital.com/industrial/optotrakproseries-family.php

[26] (2009, April) Tracker models. [Online]. Available: http://www.ptiphoenix.com/VZmodels.php

[27] (2009, April) Rl8-g50-360 super green led specs. superbrightleds.com. [Online]. Available: http://www.superbrightleds.com/specs/g_8mm_360_specs.htm

[28] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," vol. 1, 1999, pp. 666–673 vol.1.

[29] R. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *Robotics and Automation, IEEE Journal of [legacy, pre - 1988]*, vol. 3, no. 4, pp. 323–344, 1987. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1087109

[30] R. P. Horaud, B. Conio, O. Leboulleux, and B. Lacolle, "An analytic solution for the perspective 4-point problem," *Computer Vision, Graphics and Image Processing*, vol. 47, pp. 33–44, 1989.

[31] *ISO/IEC 9899:1999 Programming languages – C*, ISO/IEC Std., 1999.

[32] *ECMA-334 C Sharp Language Specification*, ECMA International Std., Rev. 4, June 200.

[33] W. Barfield and C. Hendrix, "The effect of update rate on the sense of presence within virtual environments," *Virtual Reality*, vol. 1, no. 1, pp. 3–15, June 1995.

[34] B. Watson, N. Walker, W. Ribarsky, and V. Spaulding, "Effects of variation in system responsiveness on user performance in virtual environments," *Human Factors*, vol. 40, pp. 403–414, 1998.

[35] C. D. Wickens and P. Baker, "Cognitive issues in virtual reality," pp. 514–541, 1995.

[36] H. Anton, *Calculus - A New Horizon 6th Edition*, L. Brady, Ed. John Wiley & Sons, 1999.

[37] J. B. Fraleigh and R. A. Beauregard, *Linear Algebra 3rd Edition*, 3rd ed., L. Rosatone, Ed. Addison-Wesley Publishing Company, 1995.

[38] C. J. Forne, "3-d scene reconstruction from multiple photometric images," Ph.D. dissertation, University of Canterbury, April 2007.

[39] M. Born, E. Wolf, and A. B. Bhatia, *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*, 7th ed., M. Born, E. Wolf, and A. Bhatia, Eds. Cambridge University Press, 1999.

[40] W. R. Hamilton, *Lectures on Quaternions*. Cornell University Library, 1992.

[41] T. W. B. Kibble and F. H. Berkshire, *Classical Mechanics*, 5th ed. Imperial College Press, 2004.

[42] A. J. Hanson, *Visualizing Quaternions*. Academic Press, 2006.

[43] R. Parent, *Computer Animation: Algorithms and Techniques*. Morgan Kaufmann, 2002, no. ISBN 1558605797, 9781558605794.

[44] M. S. Nixon and A. S. Aguado, *Feature extraction and image processing*. Newnes, 2002.

[45] A. A. Goshtasby, *2-D and 3-D Image Registration: For Medical, Remote Sensing, and Industrial Applications*. Wiley-IEEE, 2005.

[46] J. C. Russ, *The image processing handbook*, 5th ed. CRC Press, 2006.

[47] M. R. Shortis, T. A. Clarke, and T. Short, "Comparison of some techniques for the subpixel location of discrete target images," in *Proc. SPIE Vol. , Videometrics III*, S. F. El-Hakim, Ed., vol. 2350, no. 1. SPIE, 1994, pp. 239–250.

[48] M. Shortis, T. Clarke, and S. Robson, "Practical testing of the precision and accuracy of target image centering algorithms," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, ser. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, S. F. El-Hakim, Ed., vol. 2598, Sep. 1995, pp. 65–76.

[49] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.

[50] M. Brown and D. G. Lowe, "Unsupervised 3d object recognition and reconstruction in unordered datasets," in *3-D Digital Imaging and Modeling, 2005. 3DIM 2005. Fifth International Conference on*, 2005, pp. 56–63.

[51] B. Triggs, P. F. Mclauchlan, R. I. Hartley, and A. W. Fitzgibbon, *Bundle Adjustment – A Modern Synthesis*. Springer, January 2000, vol. 1883.

[52] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[53] R. M. Haralick, H. Joo, C. Lee, X. Zhuang, V. G. Vaidya, and M. B. Kim, "Pose estimation from corresponding point data," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 19, no. 6, pp. 1426–1446, 1989, 5 - Good review of pose estimation techniques.

[54] D. Dementhon and L. S. Davis, "Model-based object pose in 25 lines of code," in *European Conference on Computer Vision*, 1992, pp. 335–343.

[55] R. L. Carceroni and C. M. Brown, "Numerical methods for model-based pose recovery," Techn. Rept. 659, Comp. Sci. Dept., The Univ. of, Tech. Rep., 1997.

[56] L. Quan and Z. Lan, "Linear n-point camera pose determination," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 8, pp. 774–780, 1999.

[57] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: exploring photo collections in 3d," in *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*. New York, NY, USA: ACM, 2006, pp. 835–846.

[58] ——, "Modeling the world from internet photo collections," *International Journal of Computer Vision*, vol. 80, no. 2, pp. 189–210, November 2008.

[59] (2009, April) Flickr website. [Online]. Available: http://www.flickr.com/

[60] R. M. Haralick, C.-N. Lee, K. Ottenberg, and M. Nölle, "Review and analysis of solutions of the three point perspective pose estimation problem," *Int. J. Comput. Vision*, vol. 13, no. 3, pp. 331–356, December 1994.

[61] D. DeMenthon and L. S. Davis, "Exact and approximate solutions of the perspective-three-point problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 11, pp. 1100–1105, 1992.

[62] C. ping Lu, G. D. Hager, and E. Mjolsness, "Fast and globally convergent pose estimation from video images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 610–622, 2000.

[63] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, June 1981.

[64] *ADSP-BF533 Blackfin Processor Hardware Reference*, 3rd ed., July 2006. [Online]. Available: www.analog.com

[65] (2009, March) Firefly mv. WWW. Point Grey Research. [Online]. Available: http://www.ptgrey.com/products/fireflymv/fireflymv.pdf

[66] G. Welch and G. Bishop, "An introduction to the kalman filter," Department of Computer Science, University of North Carolina - Chapel Hill, Tech. Rep. TR95-041, 1995.

[67] T. A. Clarke, M. A. R. Cooper, and J. G. Fryer, "Estimator for the random error in subpixel target location and its use in the bundle adjustment," A. Gruen and H. Kahmen, Eds., vol. 2252, no. 1. SPIE, 1994, pp. 161–168.

[68] J. Grenning, "Progress before hardware," *Agile Times*, vol. 4, no. 1, 2004. [Online]. Available: http://www.objectmentor.com/resources/articles/ProgressBeforeHardware.pdf

[69] O. S. Community. (2009, January) Open computer vision library. [Online]. Available: http://sourceforge.net/projects/opencvlibrary/

[70] G. R. Bradski and A. Kaehler, *Learning OpenCV*. O'Reilly, 2008.

[71] *1149.1-2001 IEEE standard test access port and boundary-scan architecture*, IEEE Std. ISBN: 0-7381-2944-5, 2001.

[72] [Online]. Available: www.section5.ch

[73] S. Brown and Z. Vranesic, *Fundamentals of Digital Logic with VHDL Design with CD-ROM*, 2nd ed. McGraw-Hill Science/Engineering/Math, July 2005.

[74] *Nios II C2H Compiler User Guide*, Nov 2008, v1.5. [Online]. Available: http://www.altera.com/literature/ug/ug_nios2_c2h_compiler.pdf

[75] T. Lorenzen. (2006, June) Interfacing micron mt9v022 image sensors to blackfin processors. Web. Analog Devices.

[76] D. Litwiller, ""ccd vs. cmos: Facts and fiction"," *Photonics Spectra*, vol. January, 2001.

[77] *Micron - 1/3-Inch Wide-VGA CMOS Digital Image Sensor MT9V022, Rev. F*, Aptina Imaging, Inc., Dec 2006.

[78] *Micron - 1/3-Inch Wide-VGA CMOS Digital Image Sensor MT9V023, Rev. B*, Aptina Imaging, Inc., 2007.

[79] *Micron - 1/3-Inch Wide-VGA CMOS Digital Image Sensor MT9V032, Rev. B*, Aptina Imaging, Inc., March 2007. [Online]. Available: http://www.aptina.com/products/image_sensors/mt9v032c12stm/#overview

[80] *Blackfin Embedded Processor - ADSP-BF531/ADSP-BF532/ADSP-BF533, Rev. E*, Analog Devices, Inc., 2007.

[81] *High Accuracy, Ultralow IQ, 1 A, anyCAP Low Dropout Regulator ADP3338, Rev. B*, Analog Devices, Inc., 2005.

[82] (2009, March) Blackfin koop - projects - blackfinone bf532. World Wide Web. Open Source Community. [Online]. Available: http://blackfin.uclinux.org/gf/project/bf1/

[83] (2007, June) I2c-bus specification and user manual. NXP Semiconductors. [Online]. Available: http://www.standardics.nxp.com/support/documents/i2c/pdf/i2c.bus.specification.pdf

[84] *FT2232C Dual USB UART / FIFO I.C.*, 1st ed., Future Technology Devices International Ltd., 2005.

[85] S. Haykin, *Communication Systems*, 4th ed., B. Zobrist, Ed. John Wiley and Sons, Inc., 2001.

[86] *IEEE 1394-2008 - IEEE Standard for a High-Performance Serial Bus*, IEEE Std., October 2008.

[87] *Blackfin Embedded Processor ADSP-BF542/ADSP-BF544/ADSP-BF547/ADSP-BF548/ADSP-BF549, Rev. B*, Analog Devices, Inc., 2009.

[88] *Universal Serial Bus Specification*, Compaq Hewlett-Packard Intel Lucent Microsoft NEC Philips Std., Rev. 2.0, April 2000.

[89] A. Azarbayejani and A. Pentl, "Submitted to the ieee symposium on computer vision camera self-calibration from one point correspondence."

[90] (2009, April) The mathworks - matlab and simulink for technical computing. [Online]. Available: http://www.mathworks.com/

[91] K. Madsen, H. B. Nielsen, and O. Tingleff, "Methods for non-linear least squares problems (2nd ed.)," Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, p. 60, 2004.

[92] J. Schmidt and H. Niemann, "Using quaternions for parametrizing 3-d rotations in unconstrained nonlinear optimization," in *VMV '01: Proceedings of the Vision Modeling and Visualization Conference 2001*. Aka GmbH, 2001, pp. 399–406.

[93] A. Ude, "Nonlinear least squares optimisation of unit quaternion functions for pose estimation from corresponding features," in *ICPR '98: Proceedings of the 14th International Conference on Pattern Recognition-Volume 1*. Washington, DC, USA: IEEE Computer Society, 1998, p. 425.

[94] J. Deacon. (2009) Model-view-controller (mvc) architecture. [Online]. Available: http://www.jdl.co.uk/briefings/index.html#mvc

[95] K. Hoshino, F. Nielsen, and T. Nishimura, "Noise reduction in cmos image sensors for high quality imaging the autocorrelation function filter on burst image sequences," *ICGST International Journal on Graphics, Vision and Image Processing, GVIP*, vol. 07, pp. 17–24, 2007.

[96] C. D. Lovell-Smith, P. J. Bones, M. P. Hayes, R. A. Johnston, and N. B. Price, "Black spot: A prototype camera module," in *Image and Vision Computing New Zealand, 2008. IVCNZ 2008. 23rd International Conference*. IEEE, November 2008, pp. 1–6.

[97] J. Trinder, "Precision of digital target location," *Photogrammetric Engineering and Remote Sensing*, vol. 55, pp. 883–886, June 1989.

[98] M. Lourakis and A. Argyros, "The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm," Institute of Computer Science - FORTH, Heraklion, Crete, Greece, Tech. Rep. 340, Aug. 2004, available from `http://www.ics.forth.gr/~lourakis/sba`.

[99] ——. (2009, May) Sparse bundle adjustment in c/c++. [Online]. Available: http://www.ics.forth.gr/~lourakis/sba/